Novell® Partner Driver Process

Supportable Kernel Drivers Independent of Novell Schedules

Susanne Oberhauser
Project Coordinator
Strategic Partner
Engineering

June 2, 2006

Novell®

Agenda

- Customer Benefits of Partner Drivers
- The SUSE® Linux* Process
- Extension to Partner Linux Driver Process
- How to Participate

Introduction

What We Do and Why

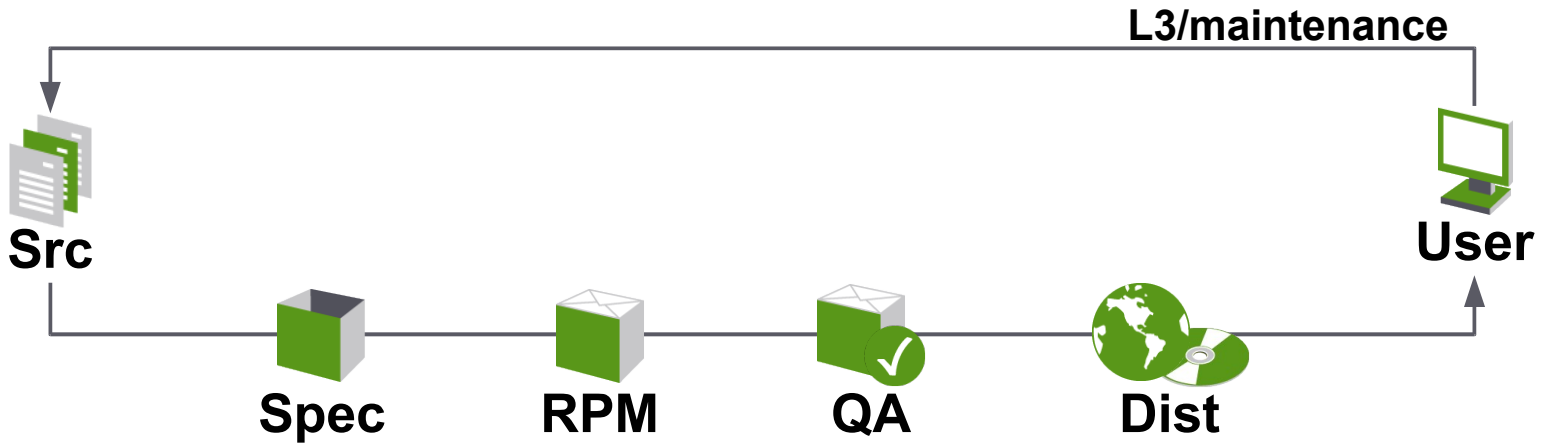
Partner Drivers Objective

Provide supported kernel drivers for SUSE® Linux independent of Novell® schedules

Examples:

- Add new filesystem
- Support additional hardware

SUSE® Linux Process



An integrated process:

development, integration, QA, distribution, support, maintenance and services

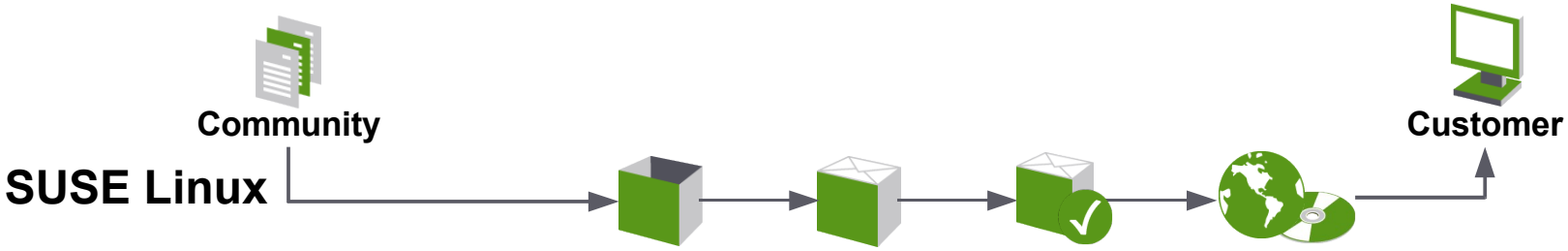
Driver Process:
Connect Novell® and Partners

- Development/backport
- Validation/QA
- Installation and updates
- Synched maintenance
- Support
- Defect resolution
- Stack certification

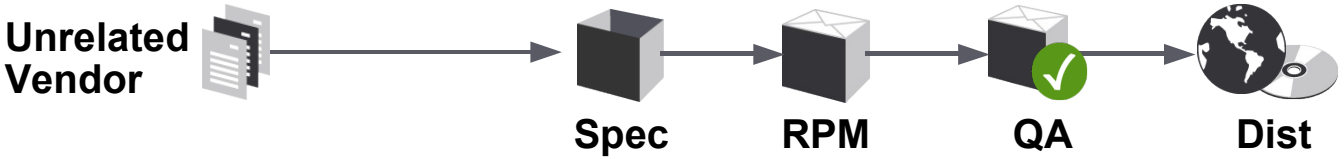
Doing It Yourself?

Why the Simple Approach Is Insufficient

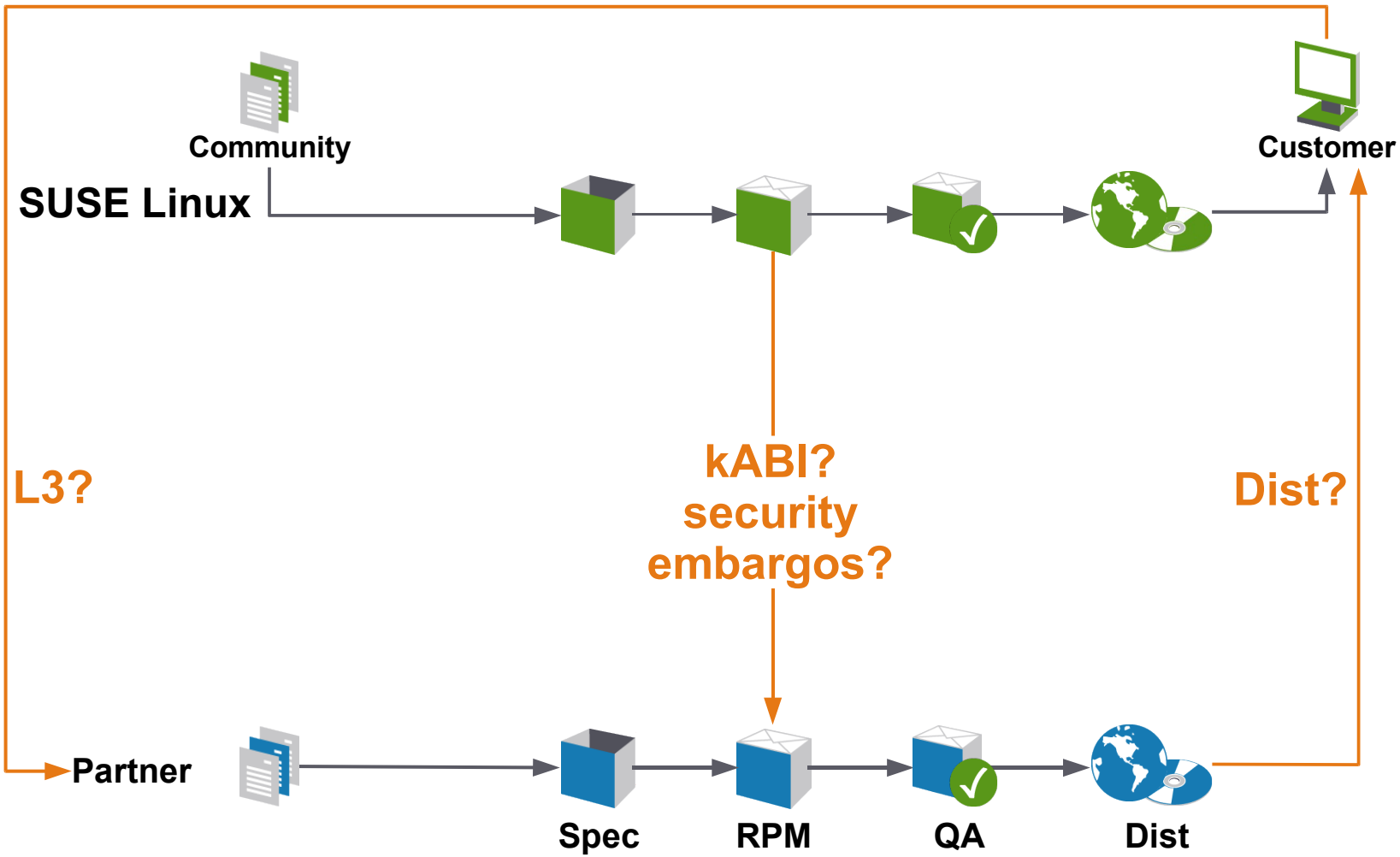
The Starting Point



- The vendor and Novell are disconnected when it comes to kernel drivers.



The Gaps of the Simple DIY Approach

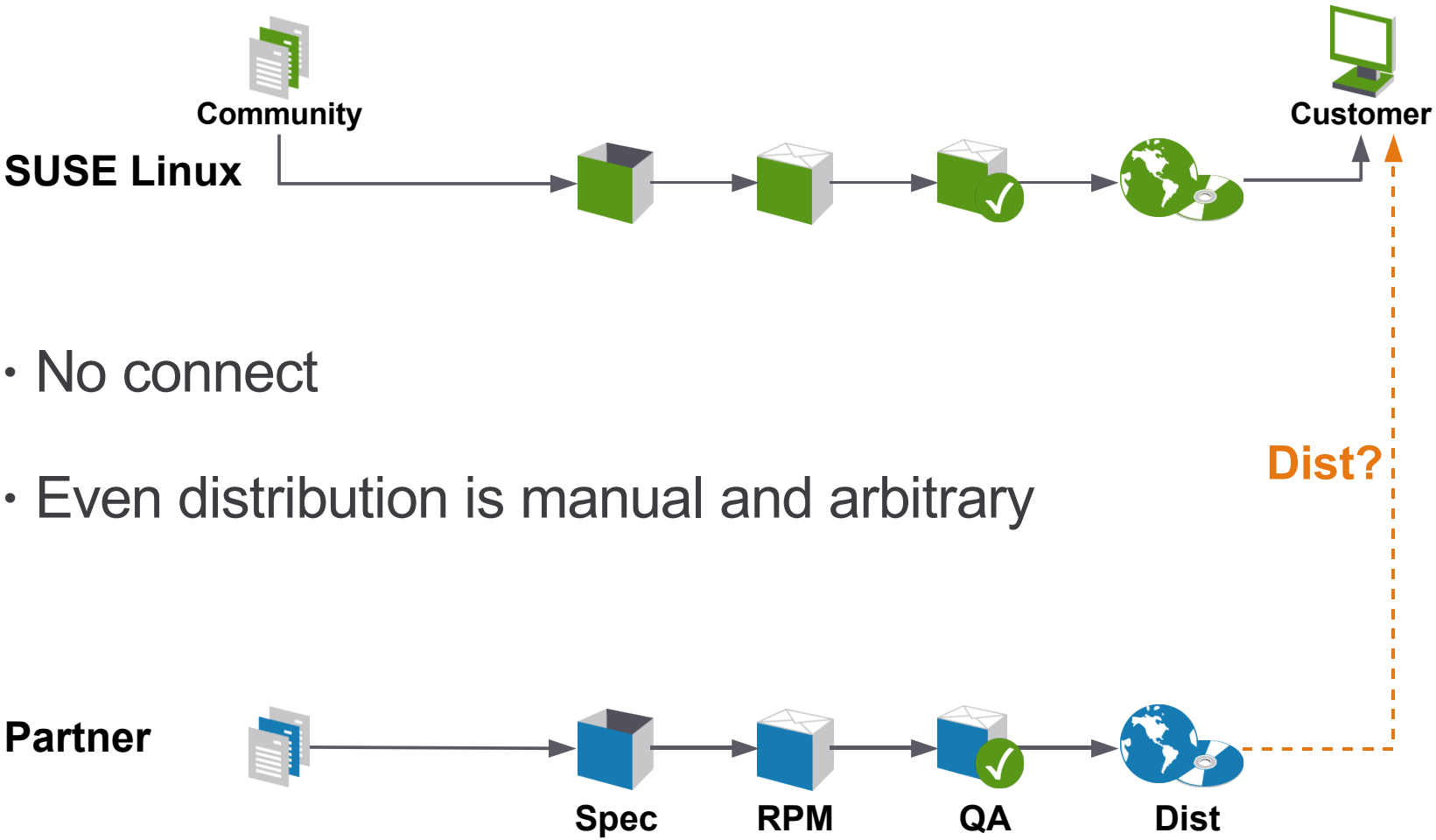


Step by Step to the Solution

How We Make It Work

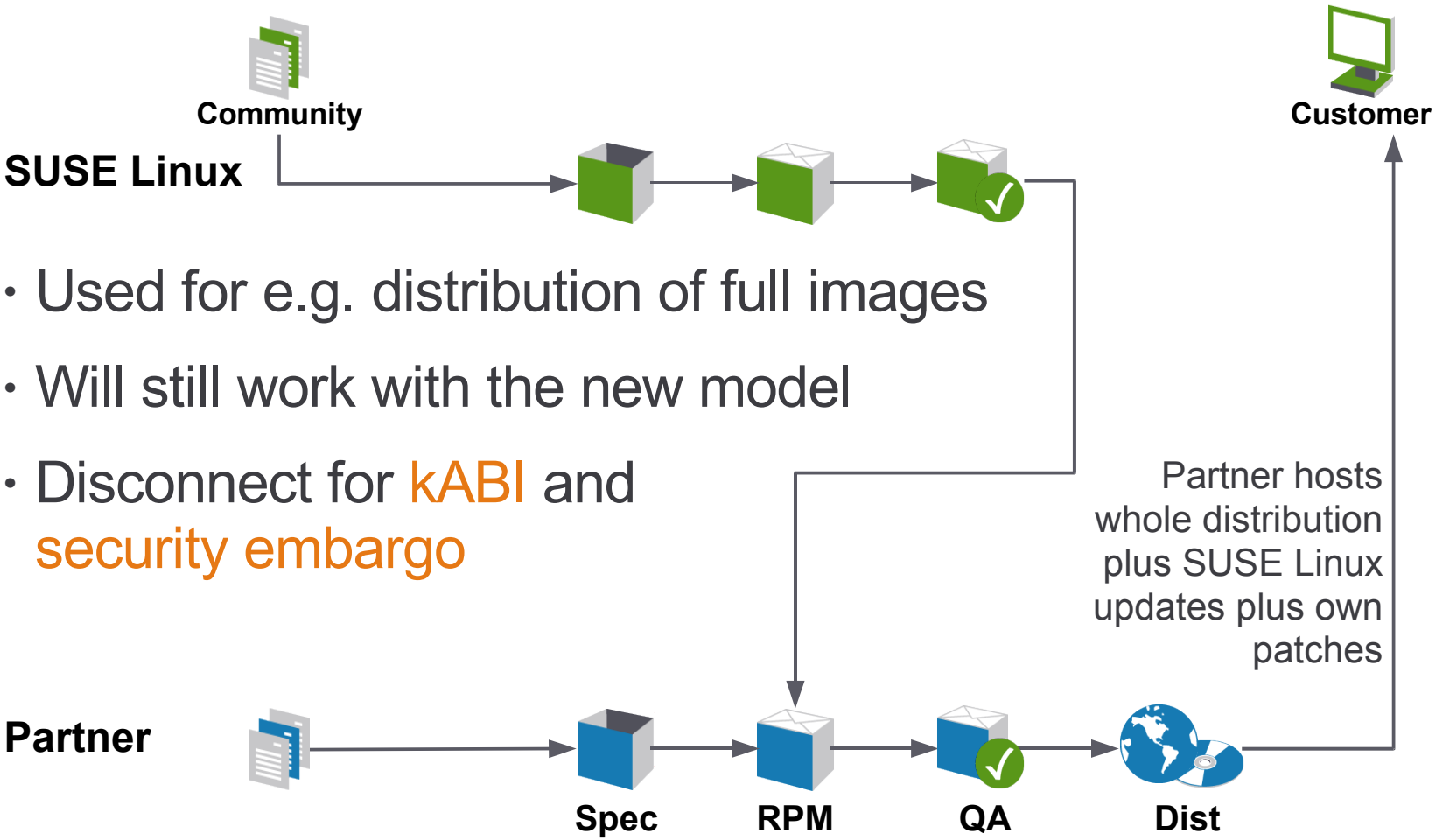
Getting Updates to the Customer

The Starting Point: No Connect



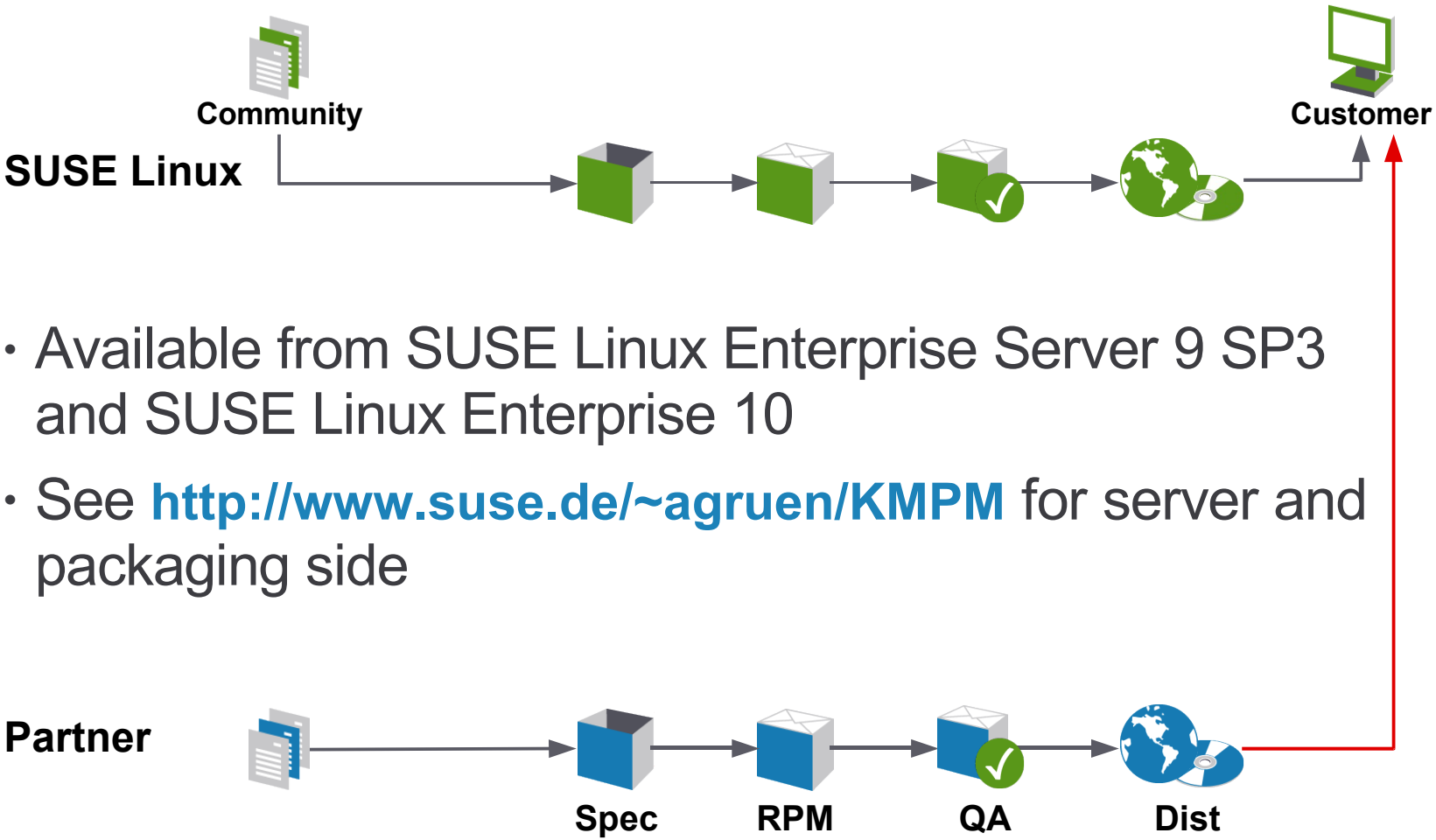
- No connect
- Even distribution is manual and arbitrary

Current Solution for Selected Partners



- Used for e.g. distribution of full images
- Will still work with the new model
- Disconnect for kABI and security embargo

New YaST Support: Update Sources



- Available from SUSE Linux Enterprise Server 9 SP3 and SUSE Linux Enterprise 10
- See **http://www.suse.de/~agruen/KMPM** for server and packaging side

Kernel Updates:

kABI, security embargos and Keeping Drivers Working

Why We Don't Promise a Fixed kABI

- Additional cost of stable kABI:
  - Kernel.org has no stable API:

    `/usr/src/linux/Documentation/stable_api_nonsense.txt`

    - Kernel refactoring becomes manageable for the community
    - Upstream modules get adapted to changes by the community
  - Some backports of new features and new drivers change kABI
  - Some immediate security fixes change kABI
  - Some bug fixes change kABI
- Autobuild: making source-compatible changes almost cost-neutral
  - Sandboxed builds are in exactly defined environments
  - kABI tool clarifies revalidation needs

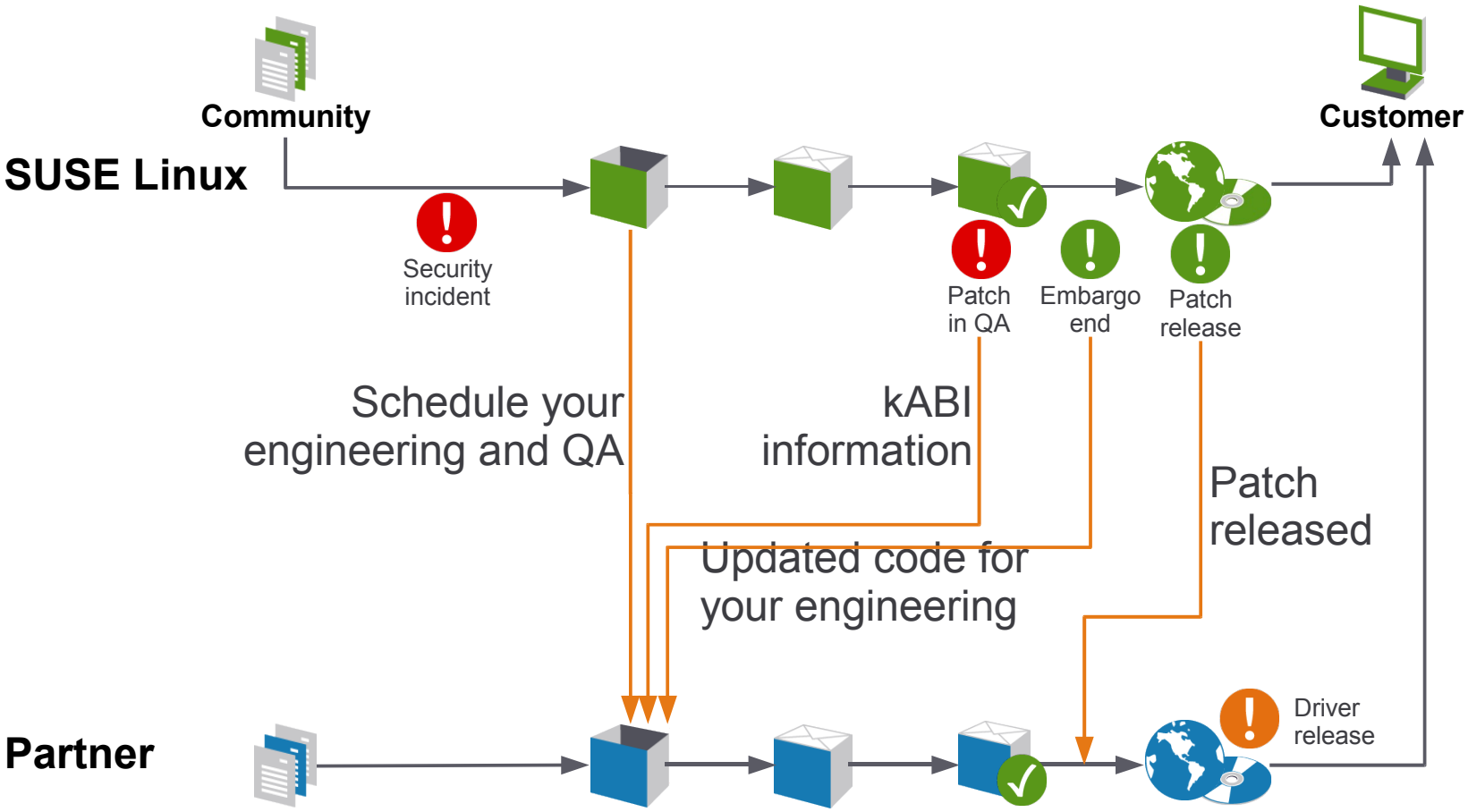
security embargos, Kernel Updates and kABI

- At incident time, a code embargo is agreed upon by the security community.
  - Code (source and binary) must be kept internal during that embargo.
  - Usually during the embargo period, Novell starts QAing the patch.
  - The patch is released at the embargo end or shortly after.

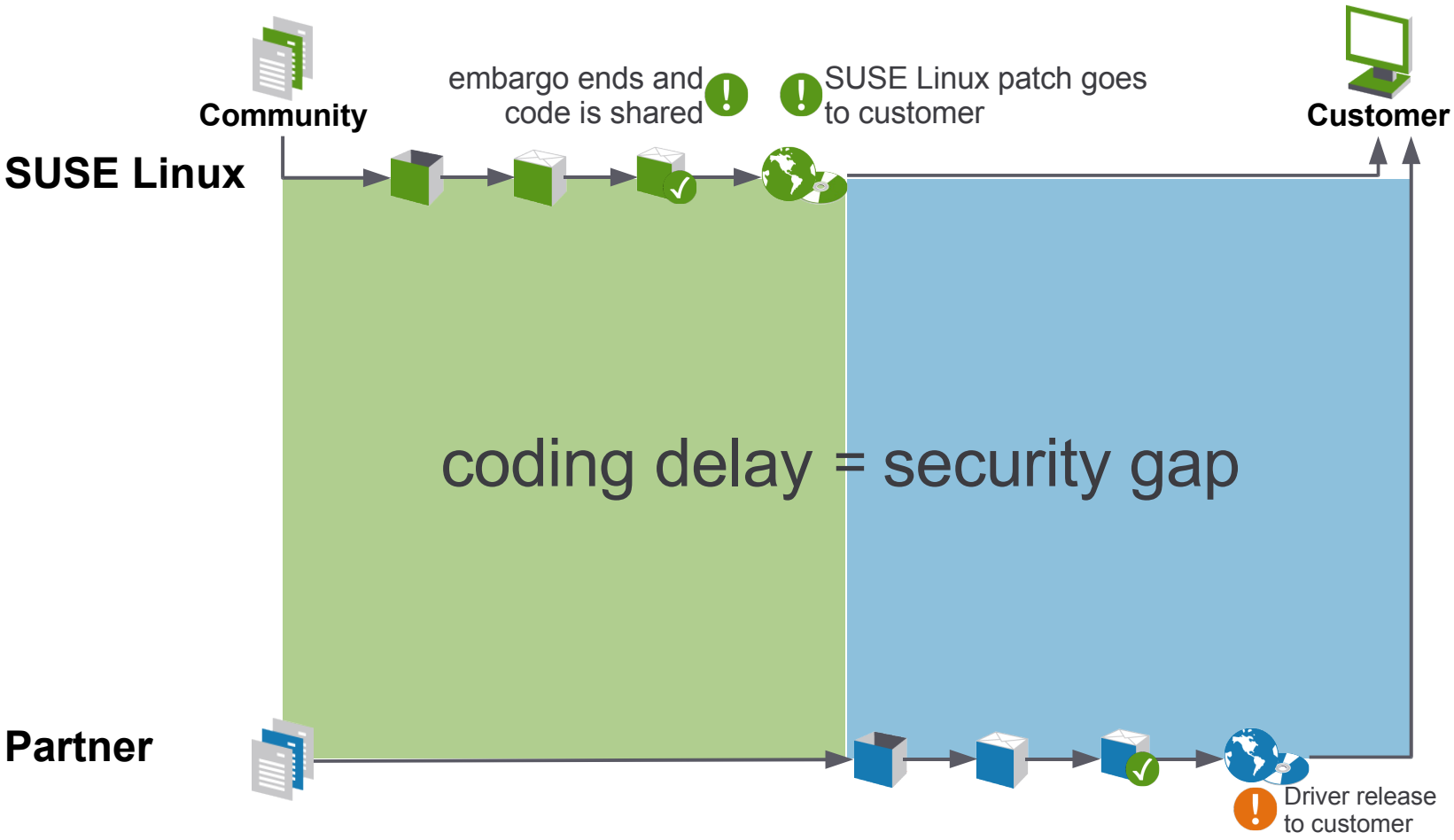
security embargos, Kernel Updates and kABI

- Novell driver process **communication** services:
  - **Notification** at security incident
    - that Novell started working on an incident
    - when we expect to release the patch
  - **Notification** when the patch goes to QA
    - which kABIs are definitely affected
  - **Updated code** when embargo ends and patch goes to QA
    - patch provided before public release for the purpose of driver rebuilds
  - **Notification** when the patch is released

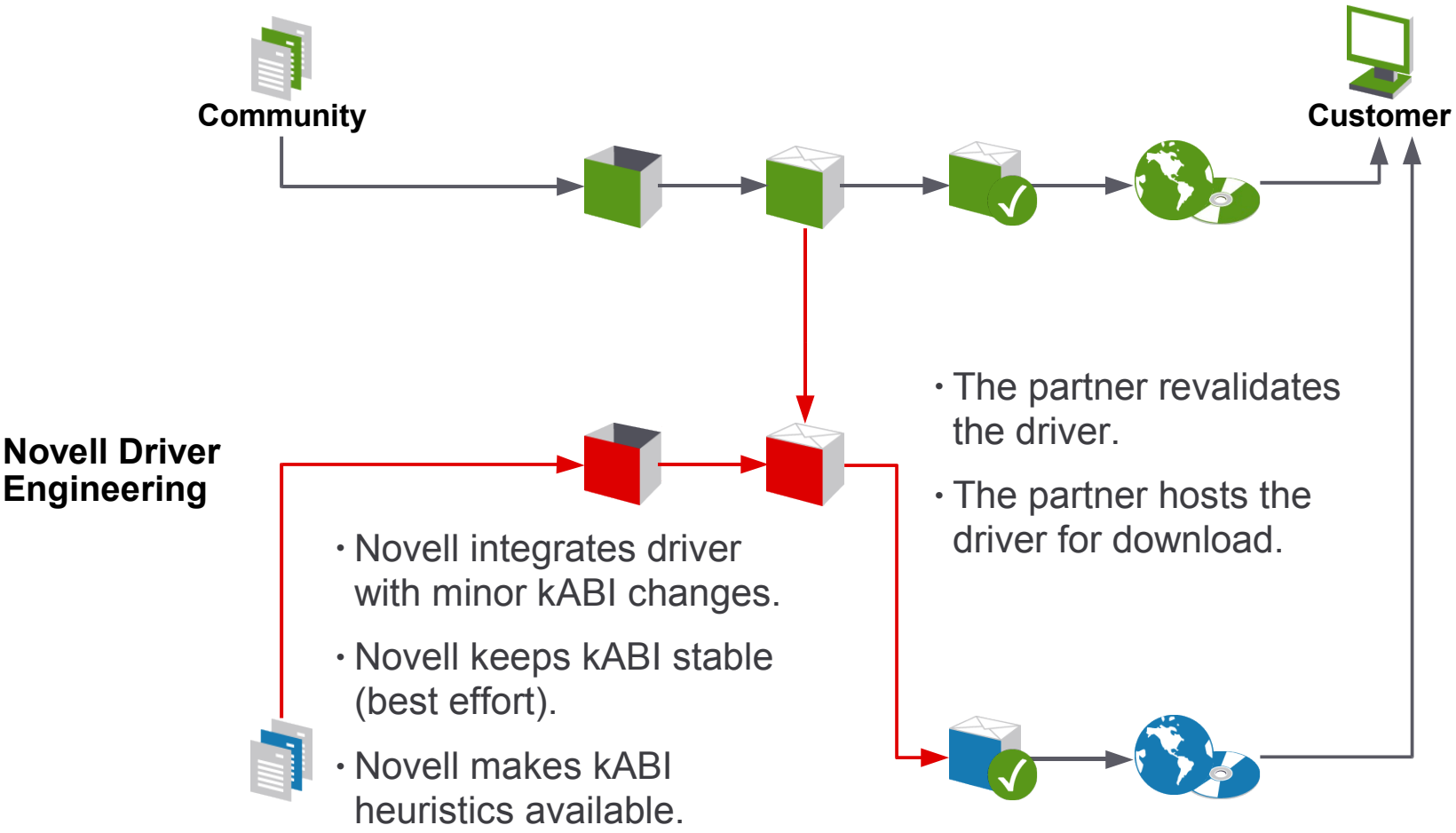
security embargos, Kernel Updates and kABI



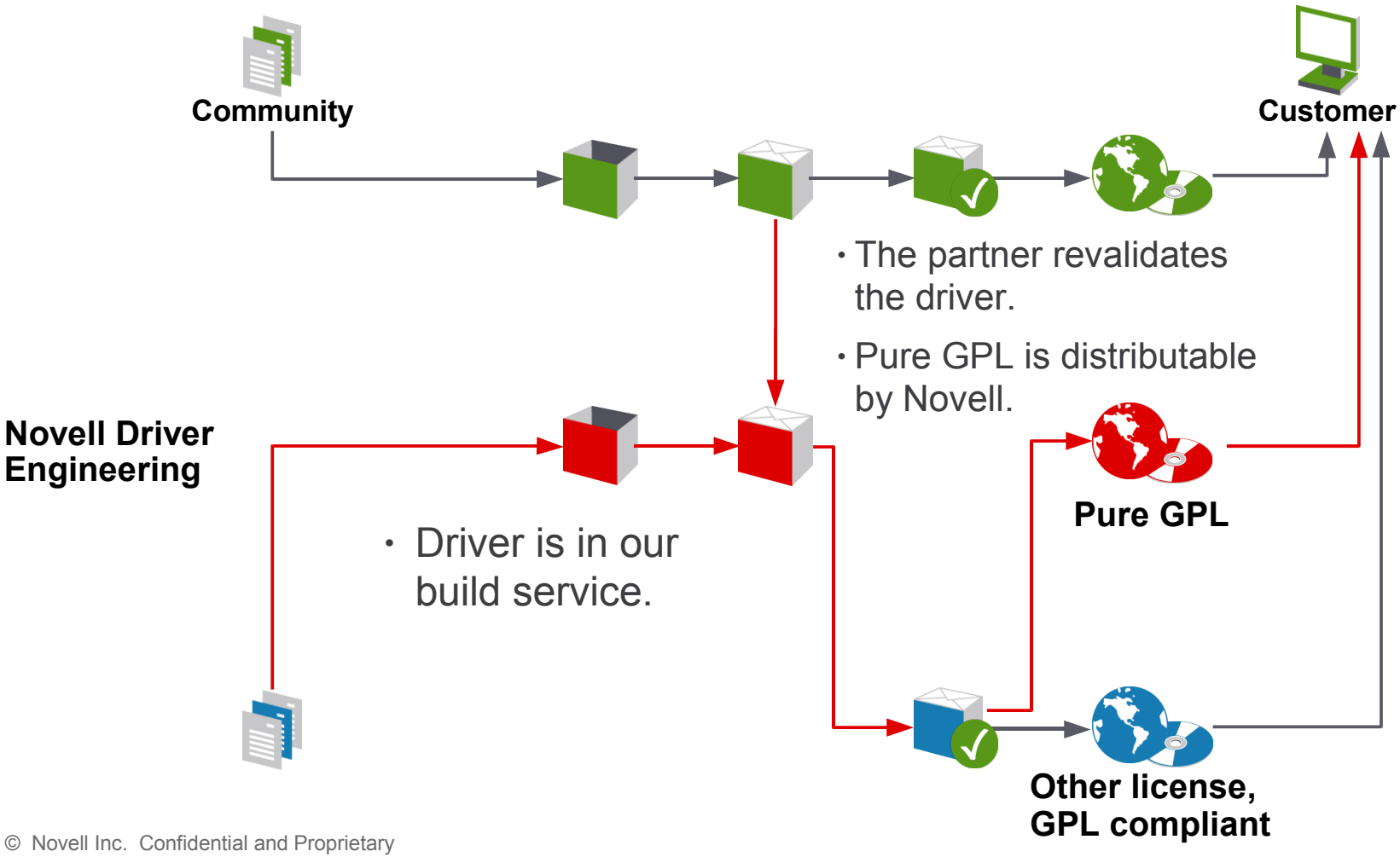
Security Gap If Code Not at Novell



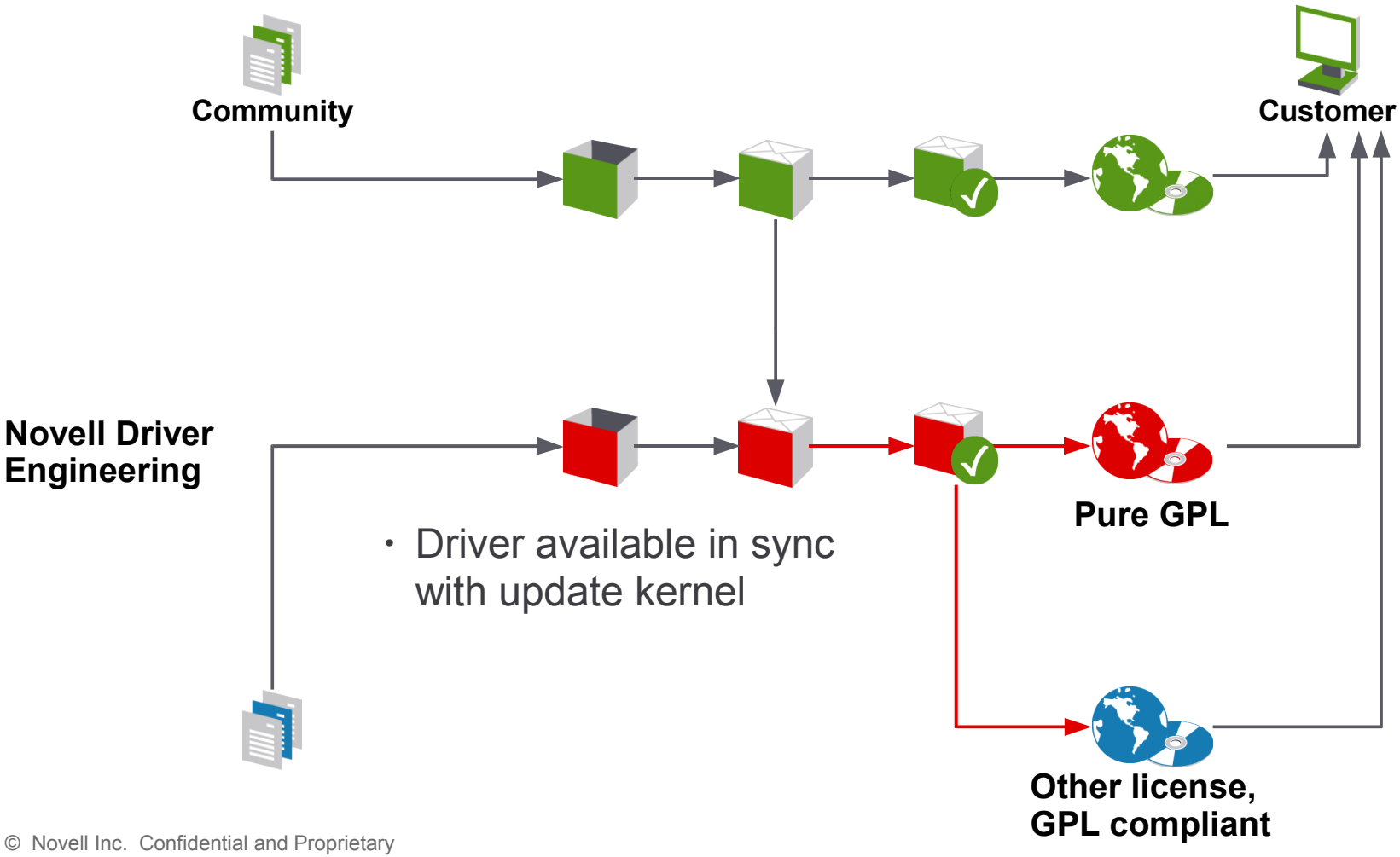
Driver Engineering Build Service



Driver Distribution Through Novell



Service: Validation by Novell

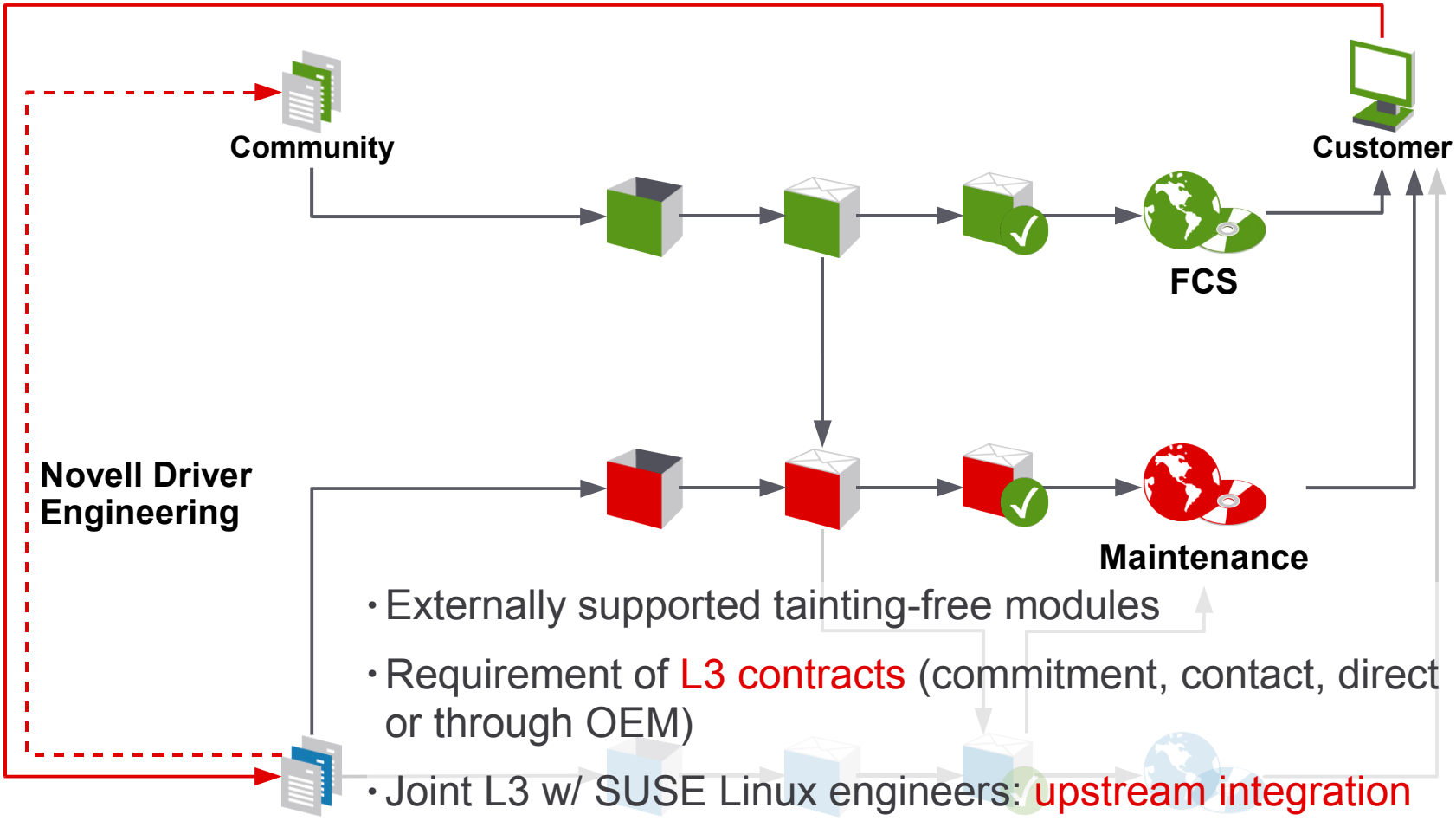


- Driver available in sync with update kernel

Joint Support and Joint Defect Resolution

The Value of Certification

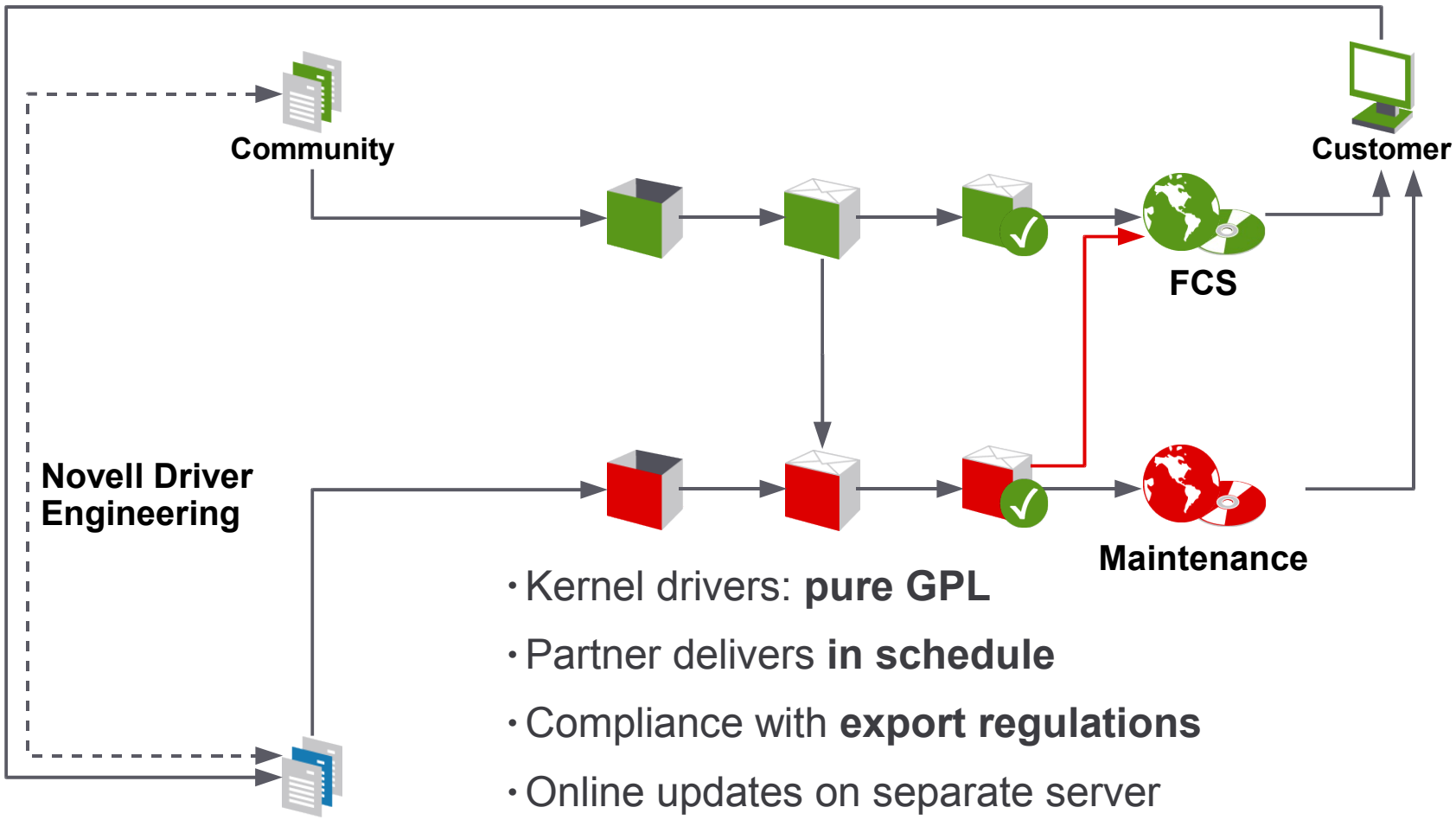
Joint Support and Defect Resolution



- Externally supported tainting-free modules
- Requirement of L3 contracts (commitment, contact, direct or through OEM)
- Joint L3 w/ SUSE Linux engineers: upstream integration
- Certification to document this to the customer

Outlook

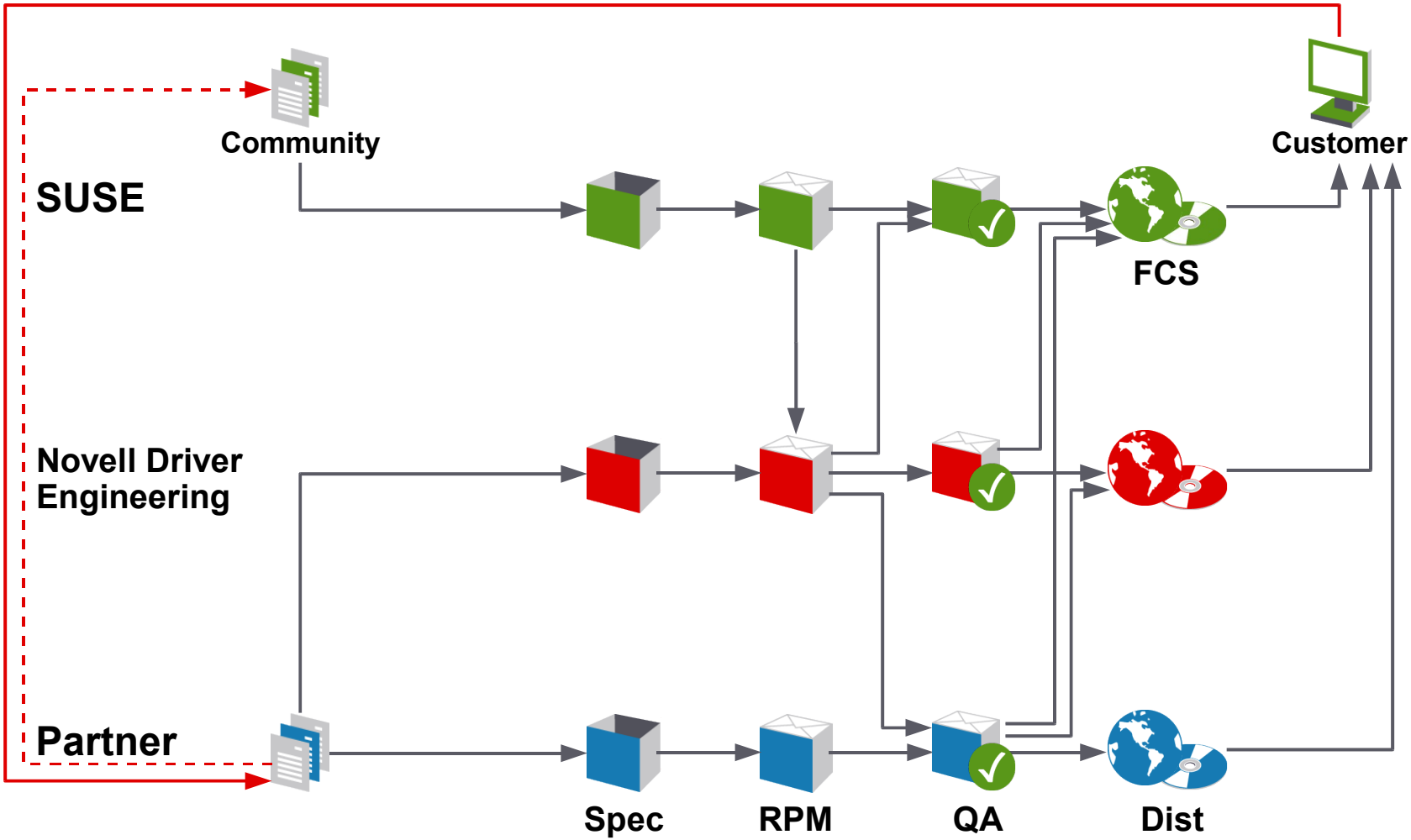
Future: Integration with Product



- Kernel drivers: **pure GPL**
- Partner delivers **in schedule**
- Compliance with **export regulations**
- Online updates on separate server

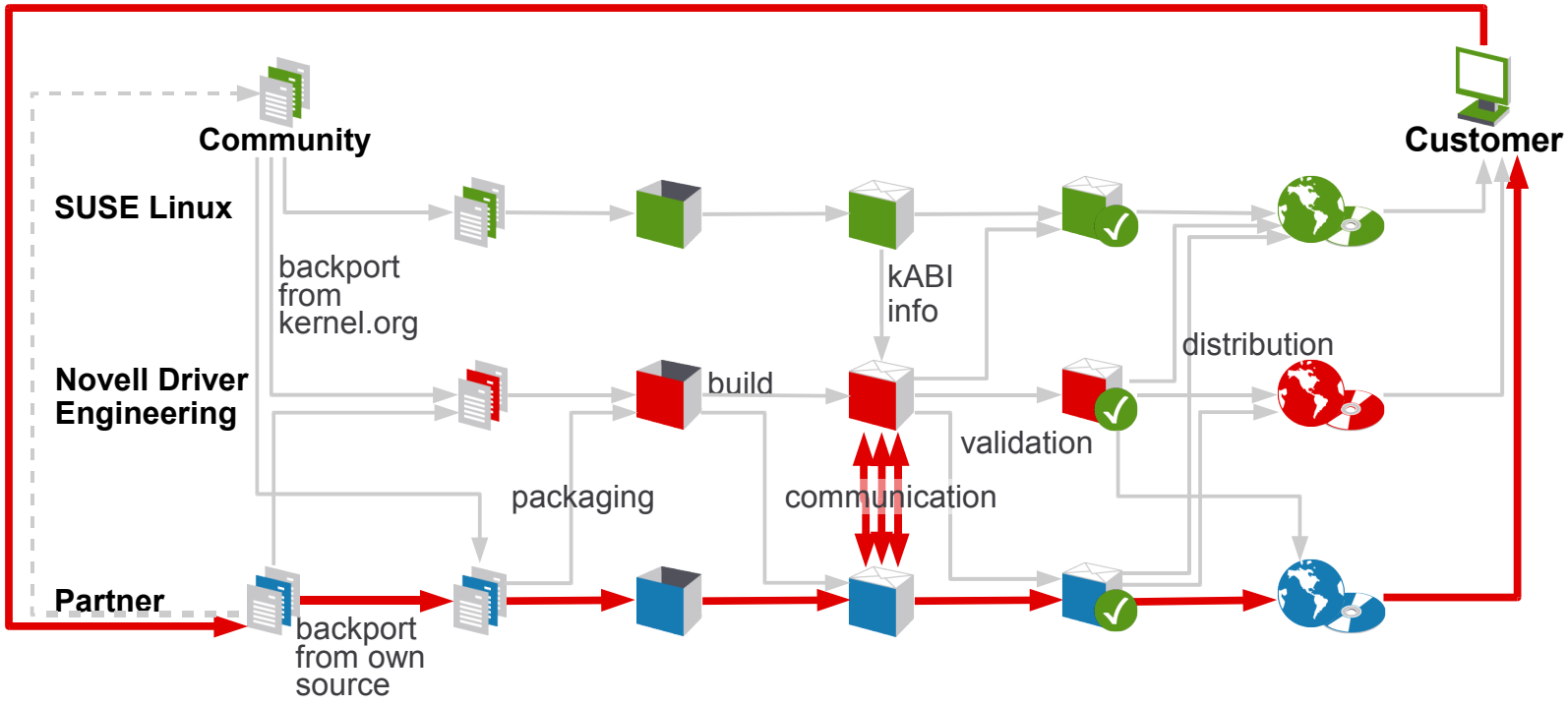
The Full Picture

Driver Process: Flexible Options



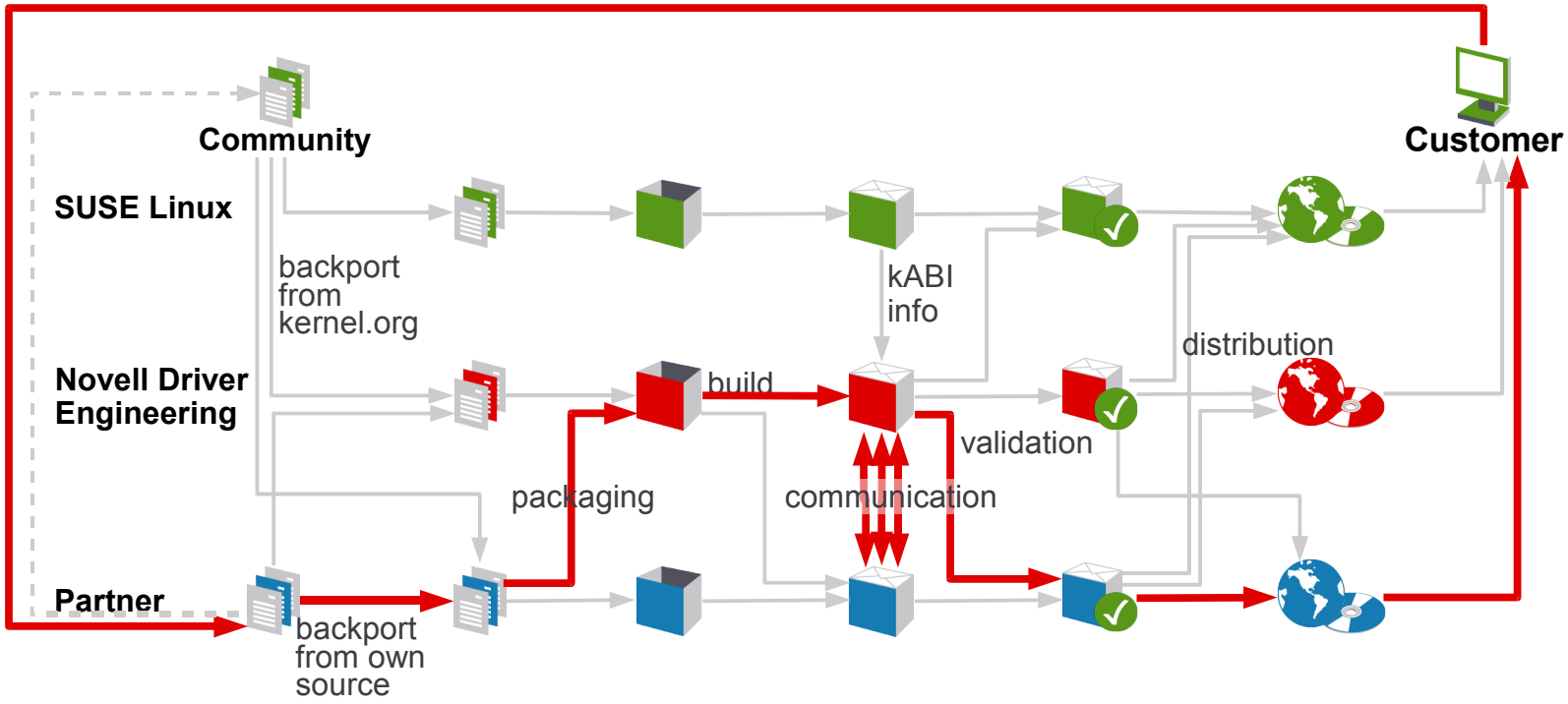
How to Participate

Driver Process Default Usage



- Partner: backport, packaging, build, validation and distribution
- Novell: kABI notifications
- Partner and Novell: support

Default for Security Sensitive Drivers

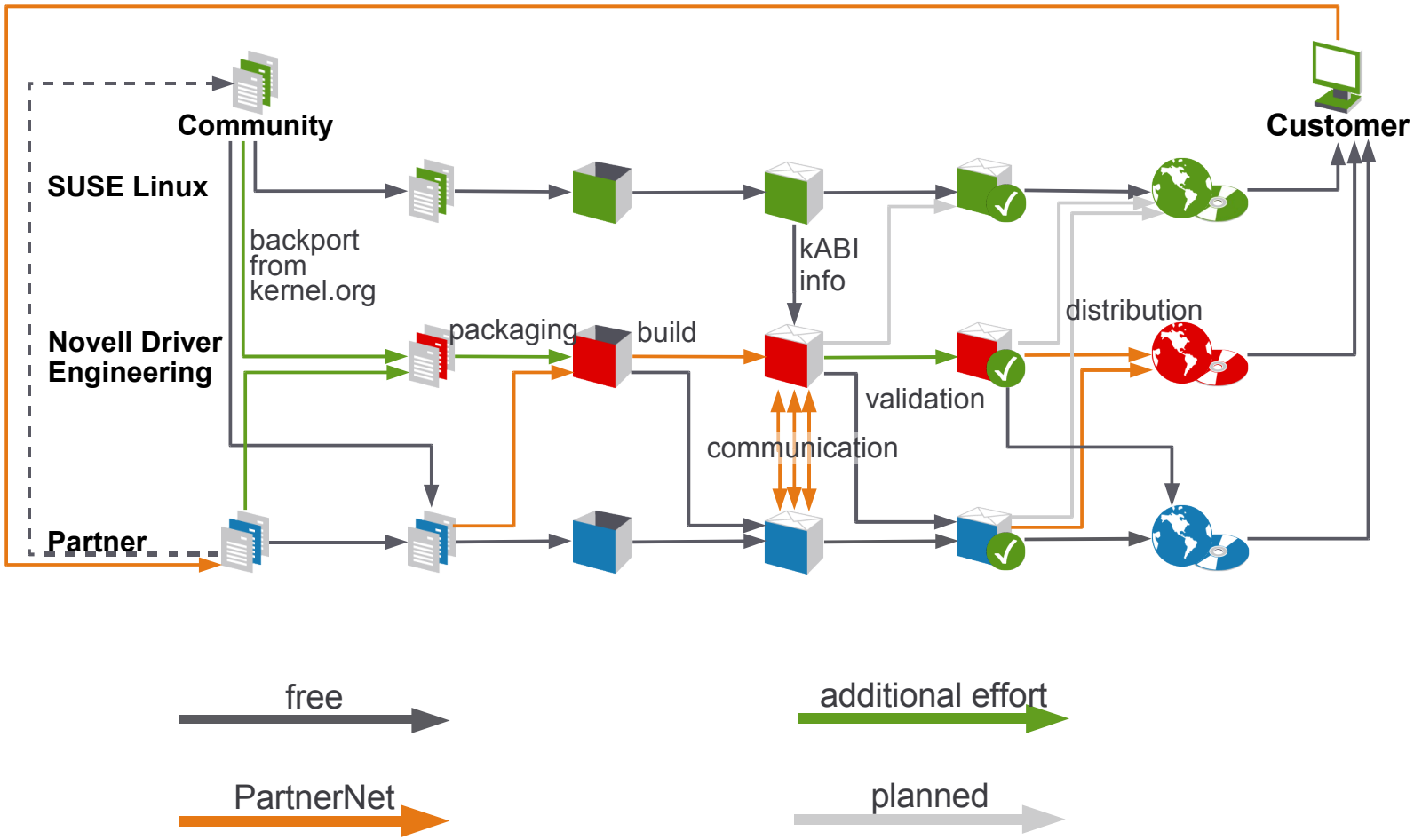


- Partner: backport and packaging
- Novell: build
- Partner: validation and distribution

- Partner and Novell: support
- Novell: kernel update notifications and rebuilds

Partnership Participation Requirements

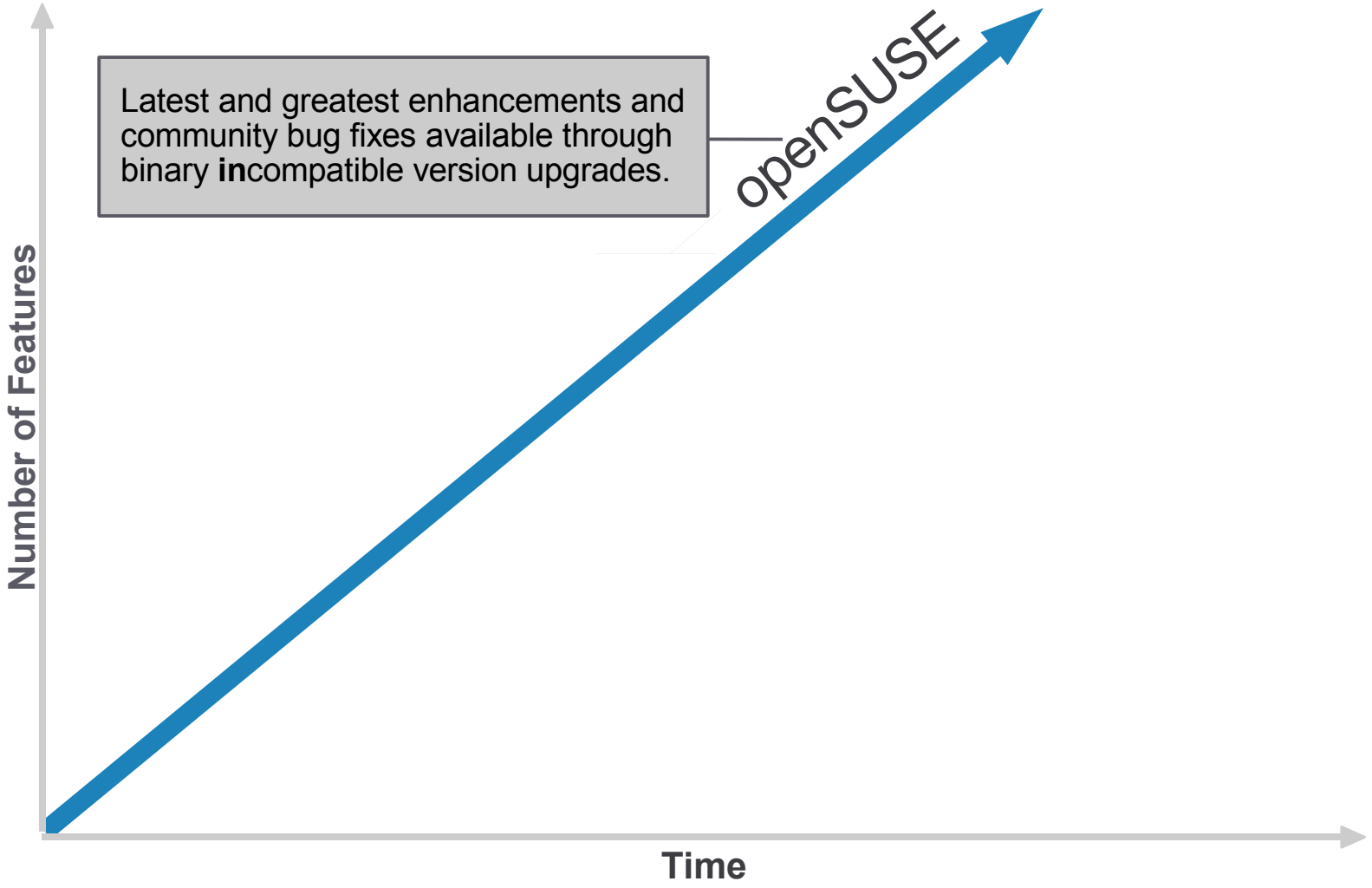
Driver Process: Participation



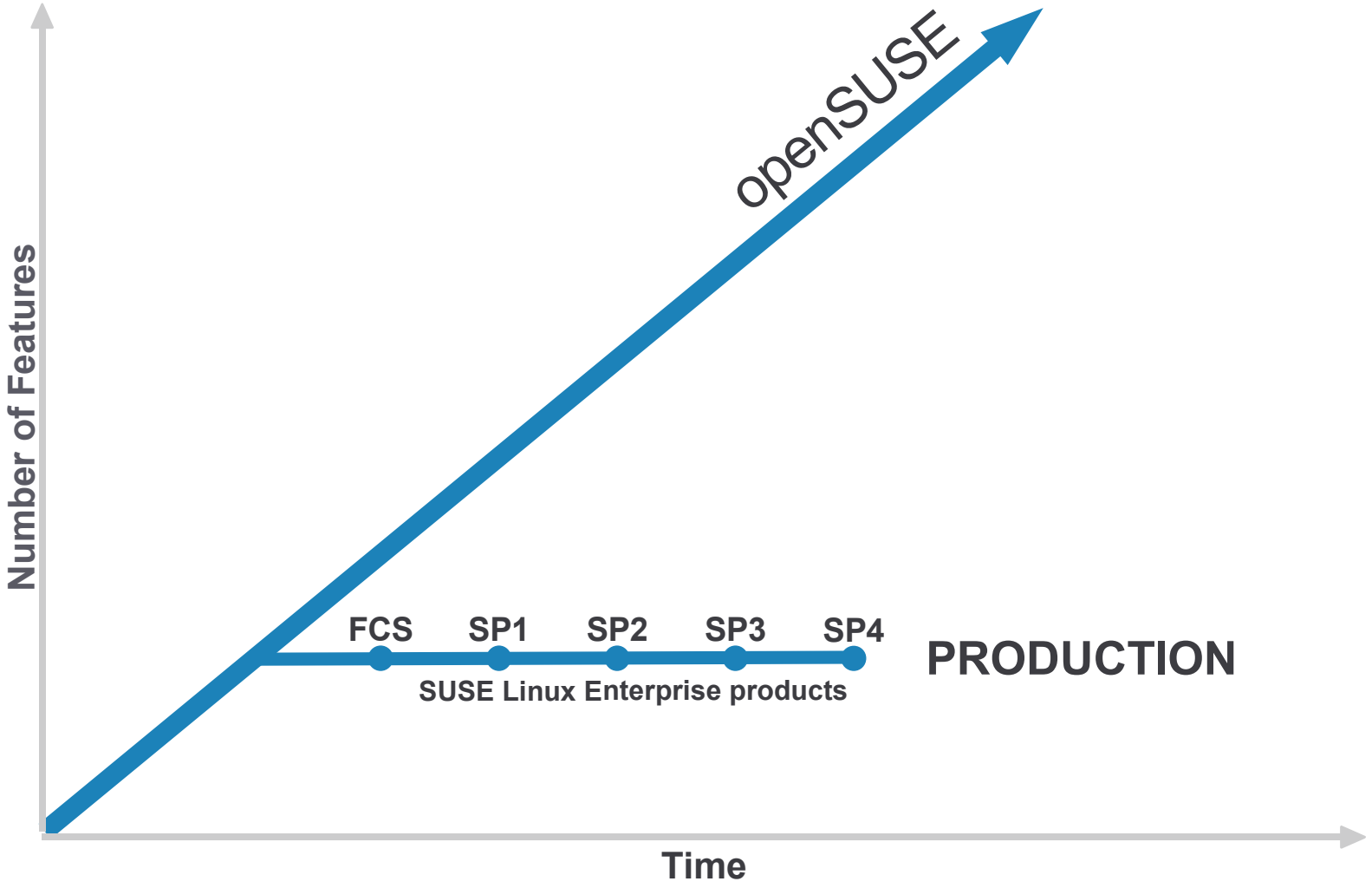
Maintenance: Backports of Fixes

Why Kernel.org Integration Is Important in but Not Sufficient for Maintenance

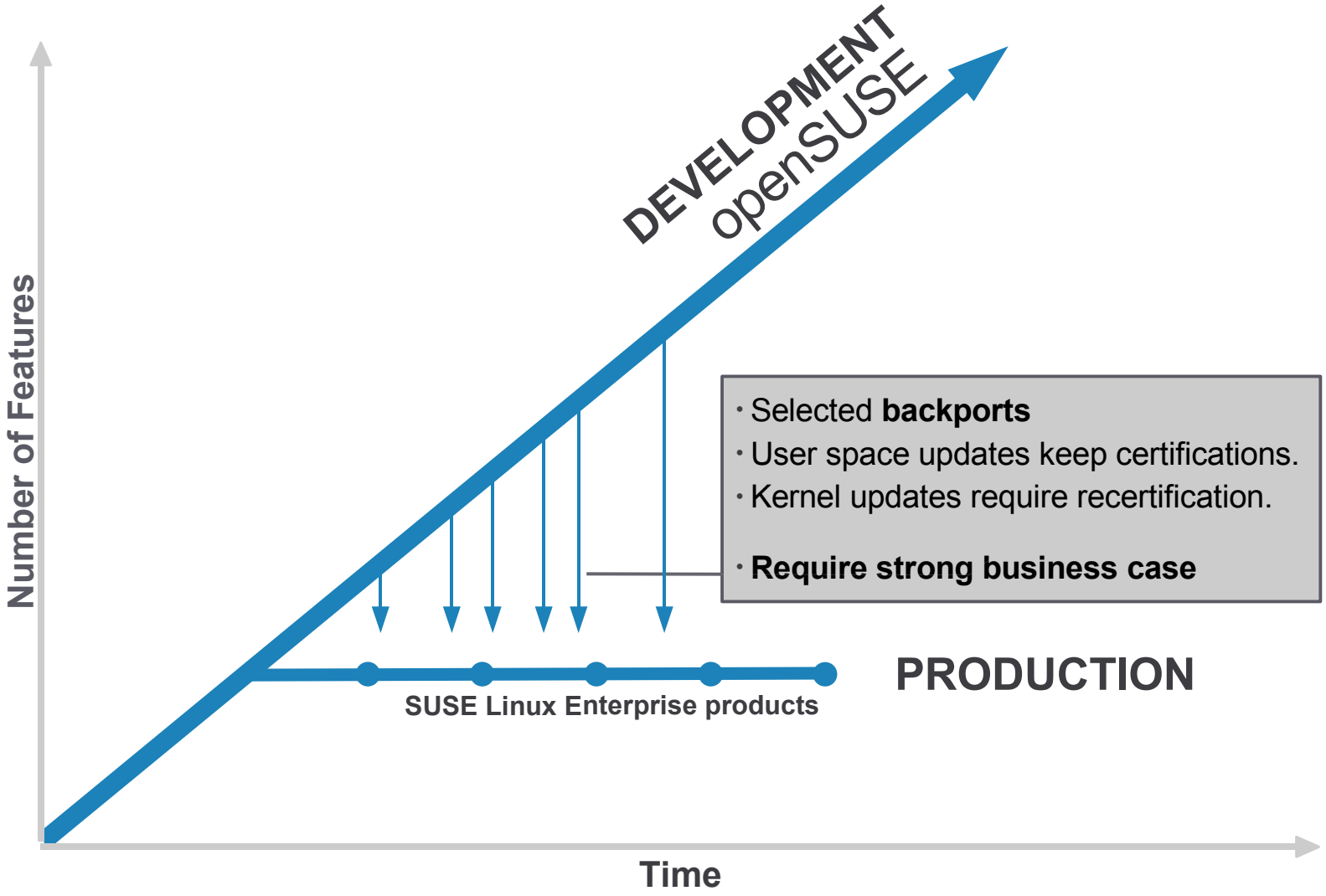
Open Source Community:
Continuous Incompatible Rebuilds



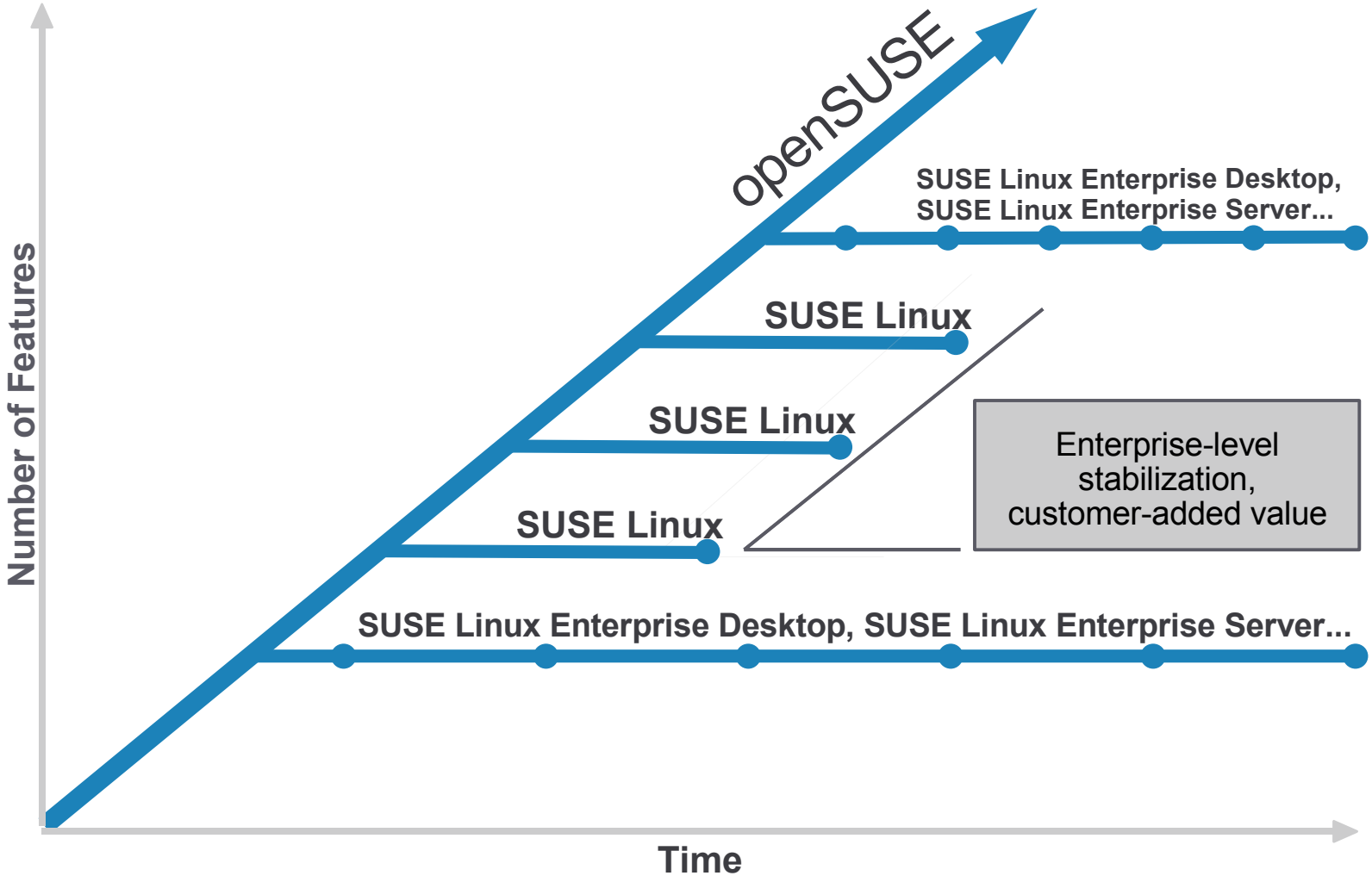
SUSE Linux Enterprise Products:
Stable User Space ABI



SUSE Linux Enterprise Products:
Only Selected Backports



SUSE Linux Enterprise Products:
Several Kernel Versions



Building, Certifying and Distributing the Drivers

Build

- General SUSE build tool:
  - build.rpm
- Driver rpm specifics:
  - http://www.suse.de/~agruen/KMPM/
- Build service
  - Suggest drivers in bugzilla.novell.com
    - Category: Partner Linux Drivers
    - Product: Partner Linux Driver Package – Code 9 or Code 10
    - Partner prepares compliant packages
    - Novell prioritizes by business relationship
  - Partner controls what is released, when and where so (NDA)

Get Certified

- Component Certification
  - In place for LAN and storage adapters
  - In the works for graphics

Distribute

- Distribution via Novell
  - Pure GPL only
  - Comply with export regulations
  - L3 relationship
  - Use build service
- Self-distribution
  - Create add-on product

Questions and Answers


Novell®

Novell® Partner Driver Process

Supportable Kernel Drivers Independent of Novell Schedules

Susanne Oberhauser
Project Coordinator
Strategic Partner
Engineering

June 2, 2006

Novell®

Welcome.

This presentation introduces you to the Novell Partner Driver Process.

With this process, Novell enables partners to provide supported drivers independent of Novell release cycles, e.g. to support newly released hardware or to add another filesystem type.

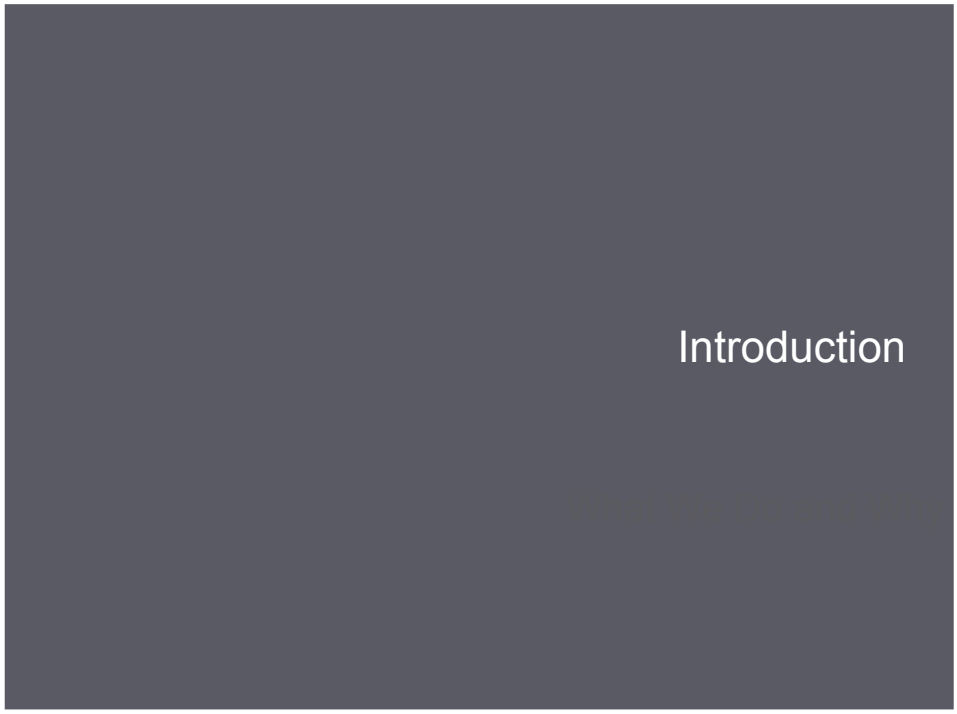
Agenda

- Customer Benefits of Partner Drivers
- The SUSE® Linux* Process
- Extension to Partner Linux Driver Process
- How to Participate

This presentation starts with what we want to achieve with the Partner Driver Process and why.

We'll then review how we distributed drivers so far and extend this SUSE Linux process into the full driver process.

When the full driver process is exposed, we'll discuss how exactly to participate.



So which customer issues do we address with the Partner Driver Process?

How have we addressed them in the past? And how do we address them in the future?

Partner Drivers Objective

Provide supported kernel drivers for SUSE® Linux independent of Novell® schedules

Examples:

- Add new filesystem
- Support additional hardware

4 © Novell Inc. Confidential and Proprietary

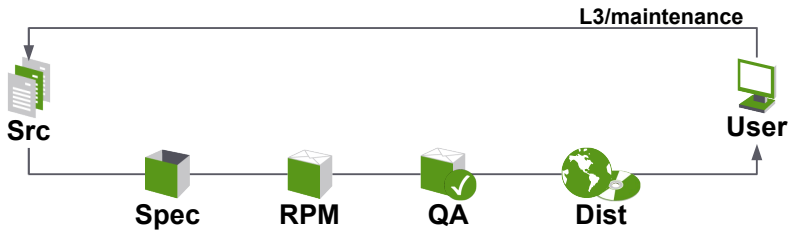
The objective is meeting needs of end customers, of Novell partners and of Novell:

Customer: get broader driver support and wider choice

Partner: become independent of Novell schedules in providing customer support for new hardware

Novell: get out of the loop for components maintained externally anyway, refocus from drivers to core kernel, offer smooth transition for code formerly distributed by SUSE Linux

SUSE® Linux Process



An integrated process:

development, integration, QA, distribution, support, maintenance and services

Getting drivers or any other packages out and providing defect resolution for them involves several interdependent steps.

This highly condensed diagram shows the SUSE Linux autobuild and maintenance process.

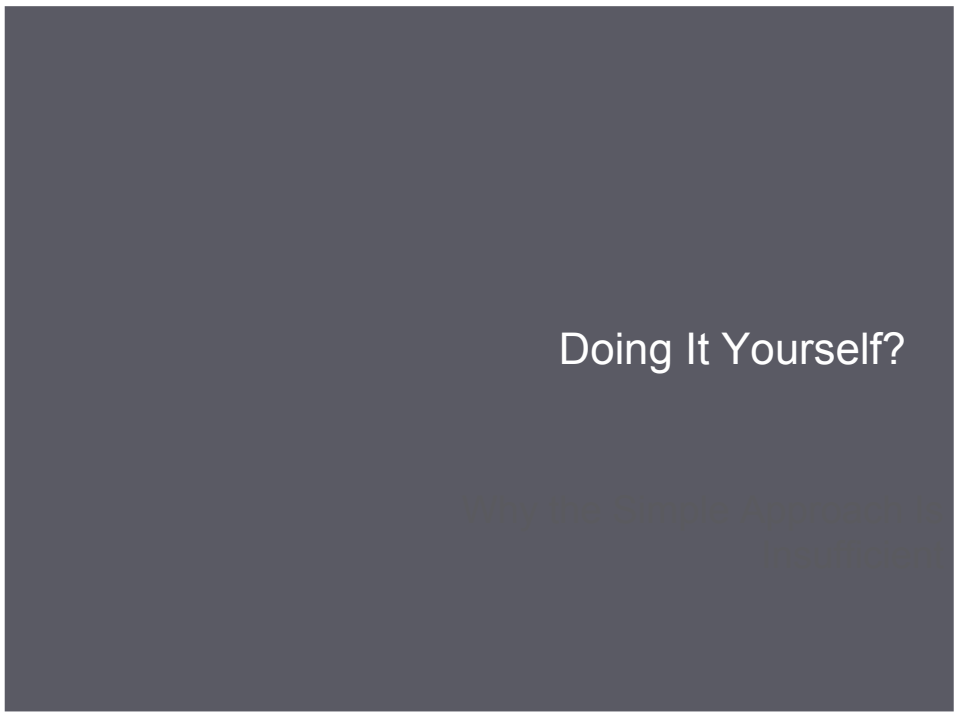
We have extended the autobuild and maintenance processes so partners can take independent ownership of their drivers.

Driver Process:
Connect Novell® and Partners

- Development/backport
- Validation/QA
- Installation and updates
- Synched maintenance
- Support
- Defect resolution
- Stack certification

To provide supportable drivers, we need to connect Novell and the partner at the process steps above.

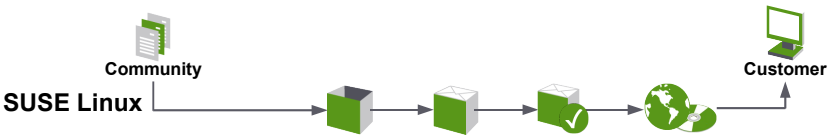
This connection plus services and technology infrastructure is the Partner Driver Process.



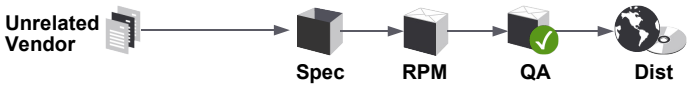
We've now talked about the process in general.

We'll next explore what we need to address, and then we'll see how we make it work.

The Starting Point



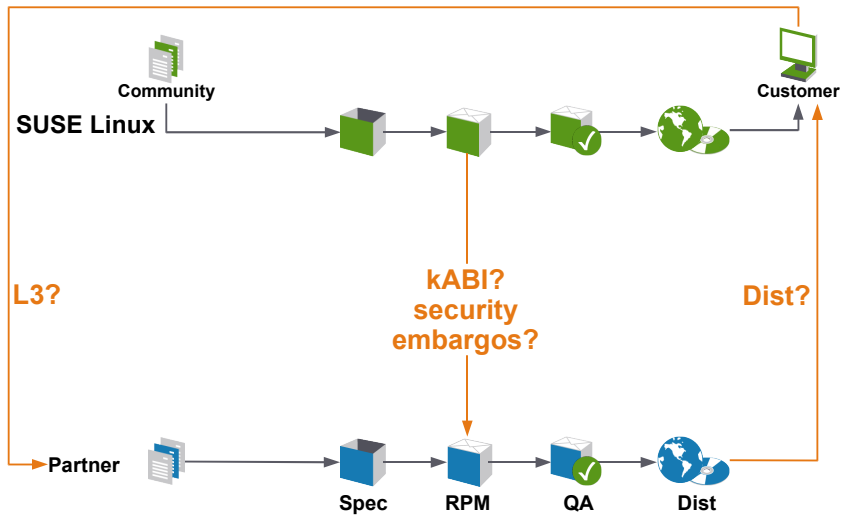
- The vendor and Novell are disconnected when it comes to kernel drivers.



If there is no connection between the partner and Novell, this is how far you get: you build drivers that fit some specific snapshot version of SUSE Linux.

The next slide depicts the gap...

The Gaps of the Simple DIY Approach



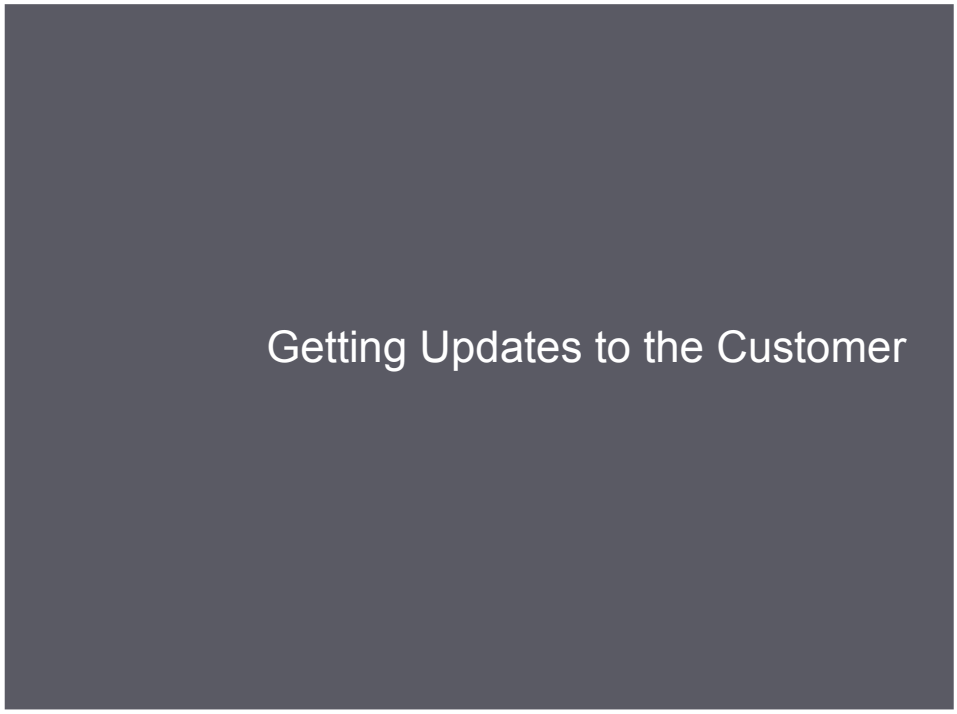
...without connect to Novell.

The partner has to figure out how to get his code to the customer alongside and in sync with our code

The partner and thus the customer have a significant delay staying in sync with security updates because of the embargo periods

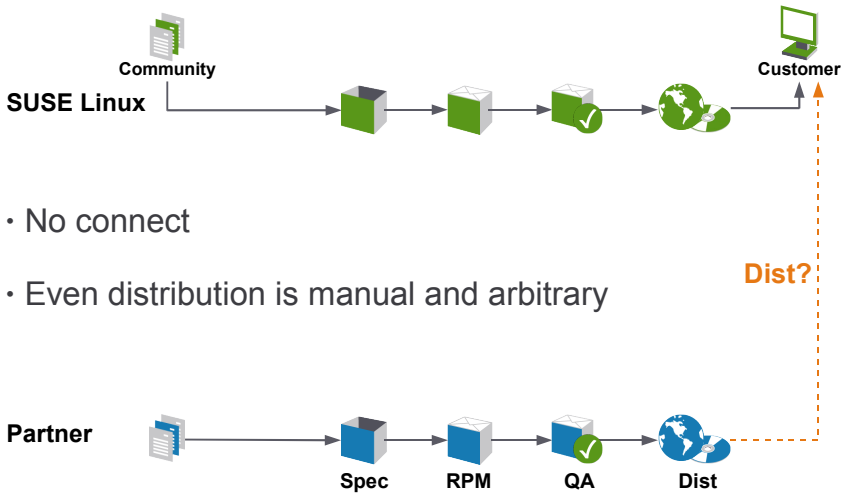
For L3 incidents, neither the partner nor Novell can help each other.

Some have used partial solutions in the past. We are now extending these partial solutions into an integrated offering.



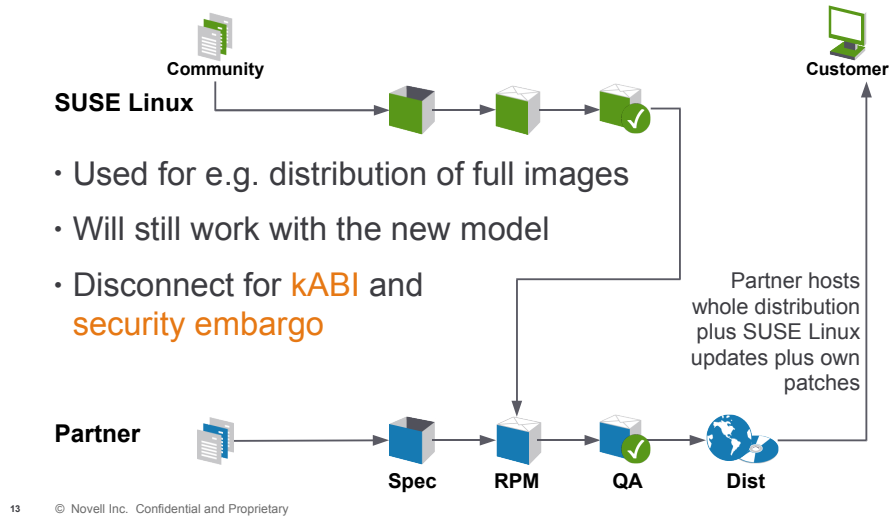
Some have used partial solutions in the past. We are now extending these partial solutions into an integrated offering.

The Starting Point: No Connect



So the first challenge is how to bring update kernels and kernel loadable to customer systems in sync so a kernel update finds matching custom modules.

Current Solution for Selected Partners



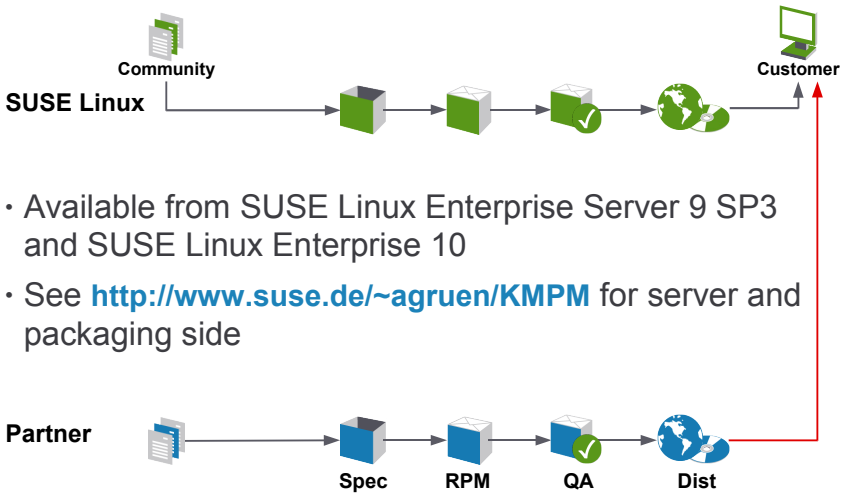
This current solution exists with selected partners:

The partner gets our updates and merges them with its code to distribute them together from its site.

Disadvantages:

- host whole SUSE Linux distro and all updates
- some delay for kABI changes
- significant delay for security updates (embargo)

New YaST Support: Update Sources



Starting with version 9, SP3 and with SUSE Linux Enterprise 10, YaST will support several update sources and cross-check dependencies so that they interact appropriately.

For example, when an update driver for the latest kernel is in the works but the update kernel is already available, YaST will wait for the updated module and will not break the old one.

The referenced documentation describes how compliant driver packages are made, how they look and how the server looks.

Kernel Updates:

kABI, security embargos and Keeping Drivers Working

Why We Don't Promise a Fixed kABI

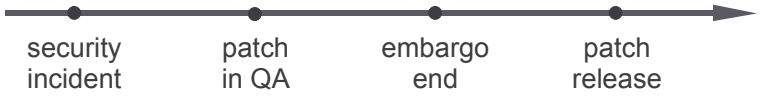
- Additional cost of stable kABI:
  - Kernel.org has no stable API:
    `/usr/src/linux/Documentation/stable_api_nonsense.txt`
    - Kernel refactoring becomes manageable for the community
    - Upstream modules get adapted to changes by the community
  - Some backports of new features and new drivers change kABI
  - Some immediate security fixes change kABI
  - Some bug fixes change kABI
- Autobuild: making source-compatible changes almost cost-neutral
  - Sandboxed builds are in exactly defined environments
  - kABI tool clarifies revalidation needs

Some claim Linux needed a fixed ABI.

A cast-in-stone kABI is simply either too expensive or too inflexible.

With the driver process, Novell reveals how we internally make our lives easier, even with fluctuating kABIs easier.

security embargos, Kernel Updates and kABI



- At incident time, a code embargo is agreed upon by the security community.
  - Code (source and binary) must be kept internal during that embargo.
  - Usually during the embargo period, Novell starts QAing the patch.
  - The patch is released at the embargo end or shortly after.

Explaining the security embargo:

when the incident occurs, an embargo end is negotiated by Linux distributors, before which no code can be shared with outside parties.

Hopefully, before that day, a security update kernel is developed and in QA.

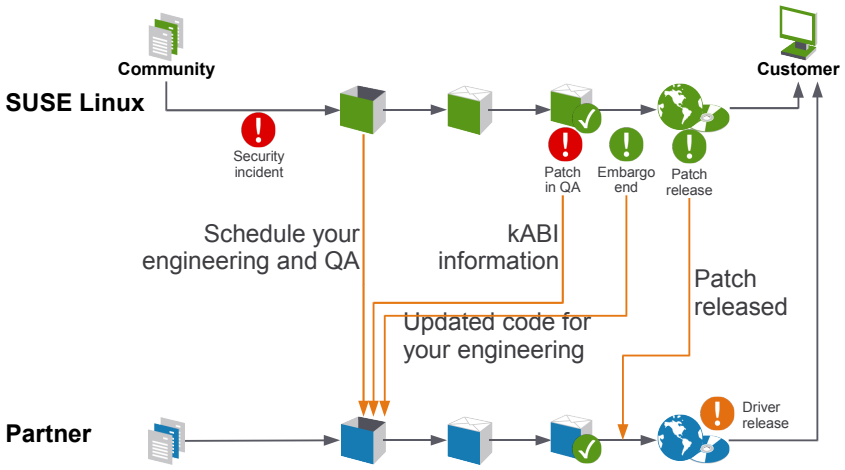
As soon as possible after the embargo ends, the updated kernel is released.

security embargos, Kernel Updates and kABI

- Novell driver process **communication** services:
  - **Notification** at security incident
    - that Novell started working on an incident
    - when we expect to release the patch
  - **Notification** when the patch goes to QA
    - which kABIs are definitely affected
  - **Updated code** when embargo ends and patch goes to QA
    - patch provided before public release for the purpose of driver rebuilds
  - **Notification** when the patch is released

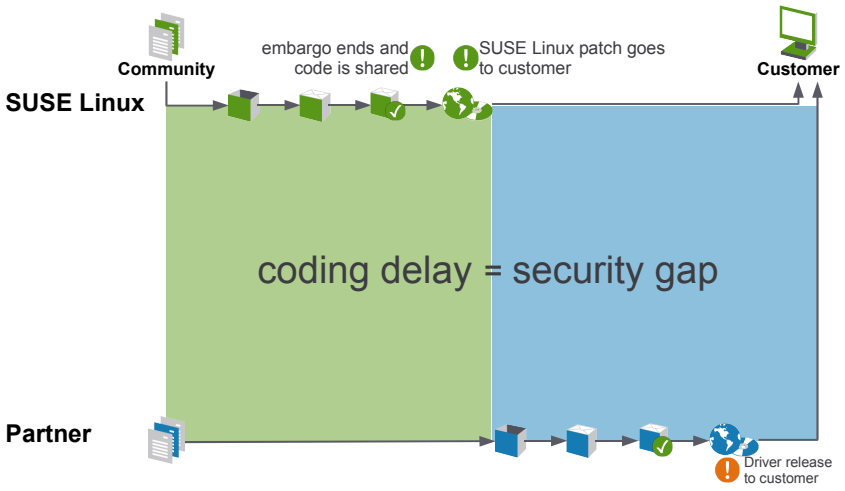
We provide kABI information to vendors involved with our process.

security embargos, Kernel Updates and kABI



This graphical visualization of the notifications leads into the next slide, which shows the delay in driver adaption to the updated kernel because of the security embargo.

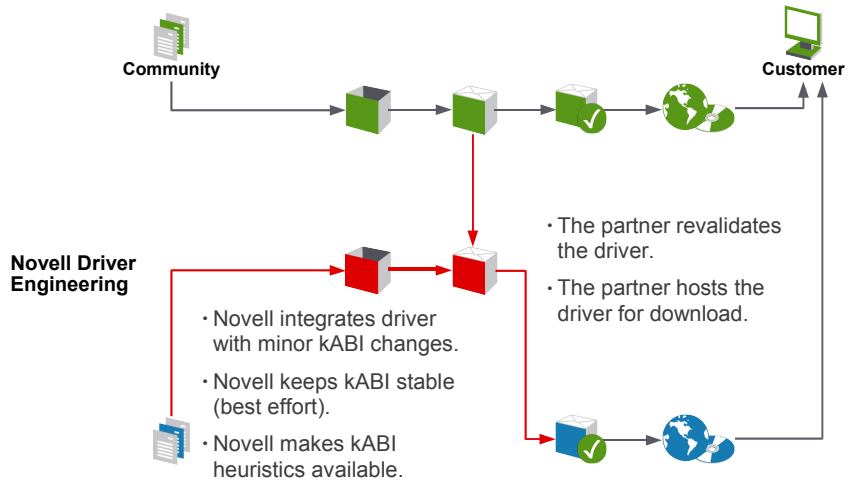
Security Gap If Code Not at Novell



Because of the notification, the partner's engineering and QA are set up to start working on the day of the embargo lift, but they may still need time to rebuild and in the worst case, adopt the driver and then validate it for public release.

This gives the security gap. Novell is closing that by (next slide) providing a build service.

Driver Engineering Build Service



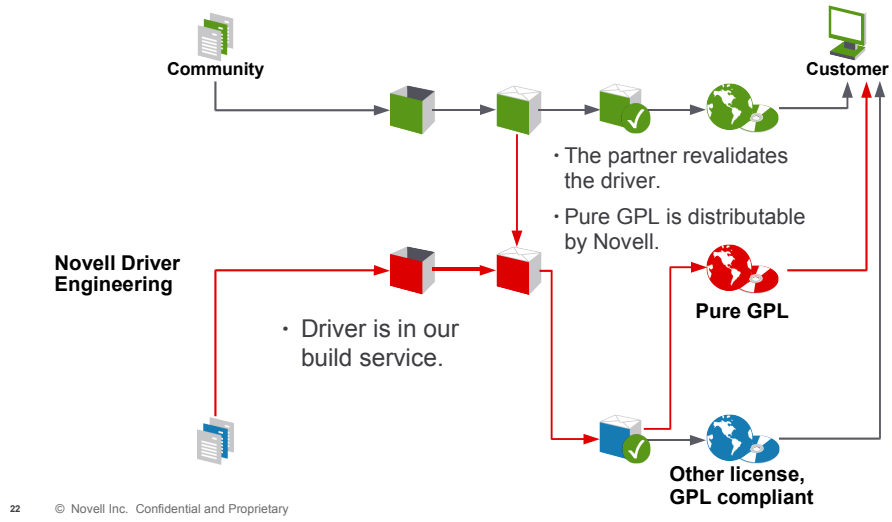
Novell Driver Engineering will get new kernels as soon as they go into SUSE Linux QA. Before the embargo is lifted, autobuild rebuilds the partner drivers within minutes.

As soon as the embargo is lifted, we can provide these to the prenotified partner QA for validation.

We will also notify the partner to let them know whether we revalidated comparable drivers.

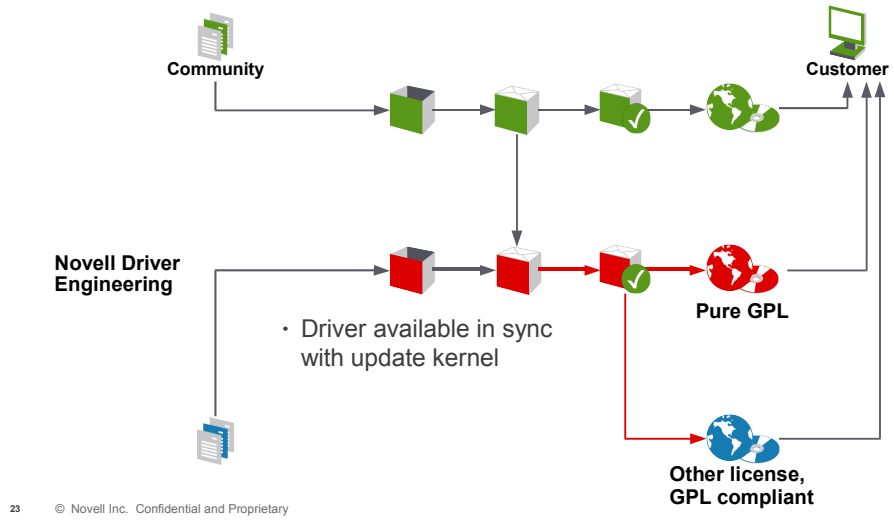
To further reduce the security gap, the partner may ask us to do the validation (next slide).

Driver Distribution Through Novell



Under specific conditions, Novell can redistribute the driver for the partner through the Novell Web site.

Service: Validation by Novell

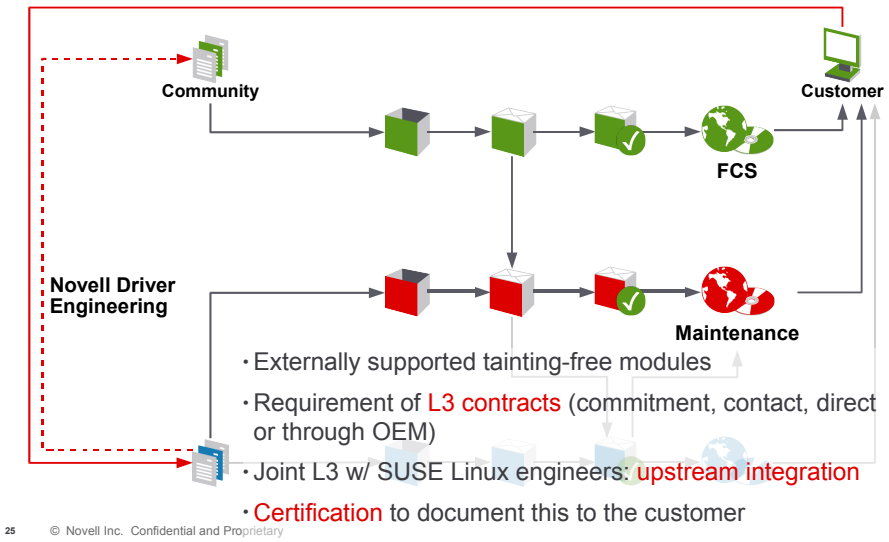


If the partner tasks us to do QA on its modules, the module can be released in sync with our kernel update. If the module works, the customer does not experience any time delay.

Joint Support and Joint Defect Resolution

The Value of Continuation

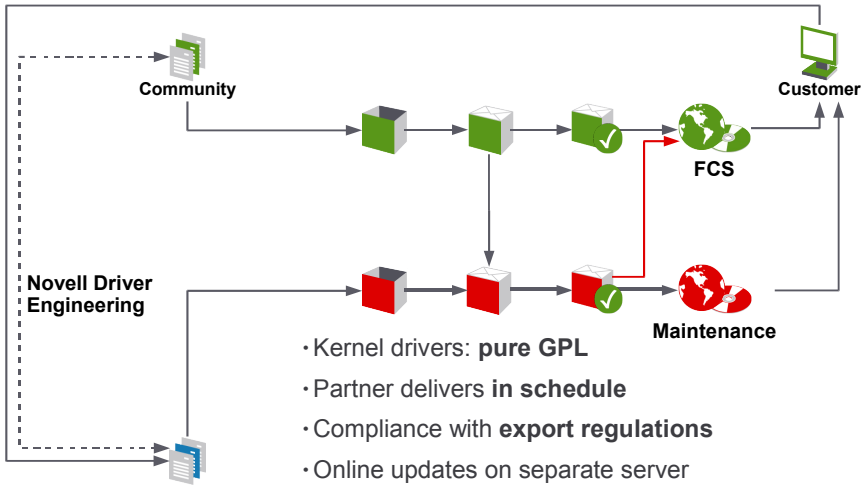
Joint Support and Defect Resolution



The alternative to individual contracts with Novell is TSANet membership.

Outlook

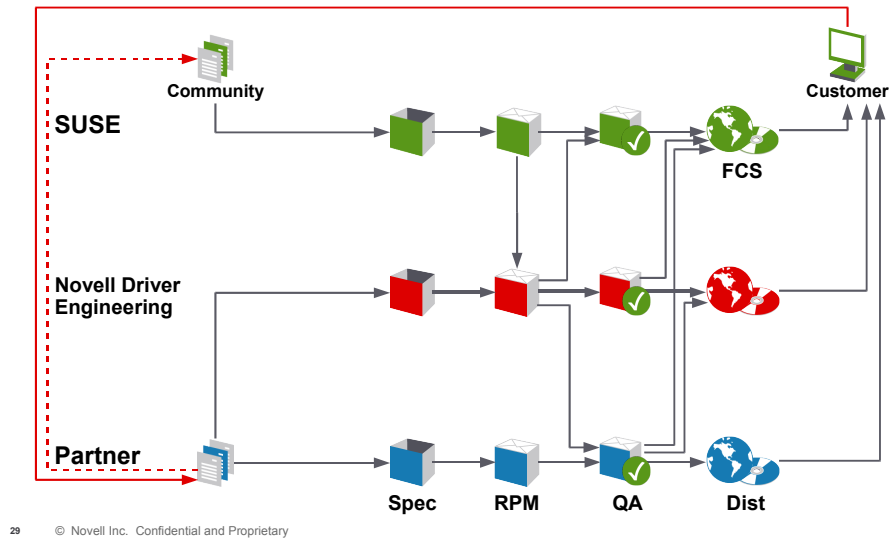
Future: Integration with Product



- Kernel drivers: **pure GPL**
- Partner delivers **in schedule**
- Compliance with **export regulations**
- Online updates on separate server

The Full Picture

Driver Process: Flexible Options



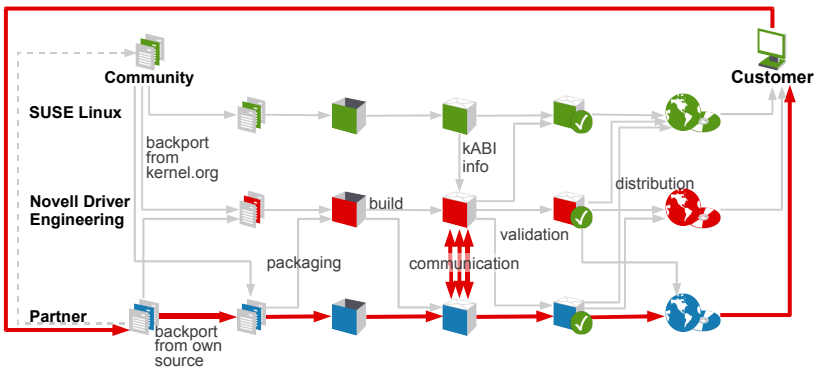
The driver process provides many options.

The flexibility of this program will also move responsibility of several steps to the owner of the driver.

We'll make a short excursion to the relationship of kernel.org and our maintained enterprise Linux kernels.

Then we'll look at the technical steps to get these drivers out to the customer.

Driver Process Default Usage



- Partner: backport, packaging, build, validation and distribution
- Novell: kABI notifications
- Partner and Novell: support

The default usage:

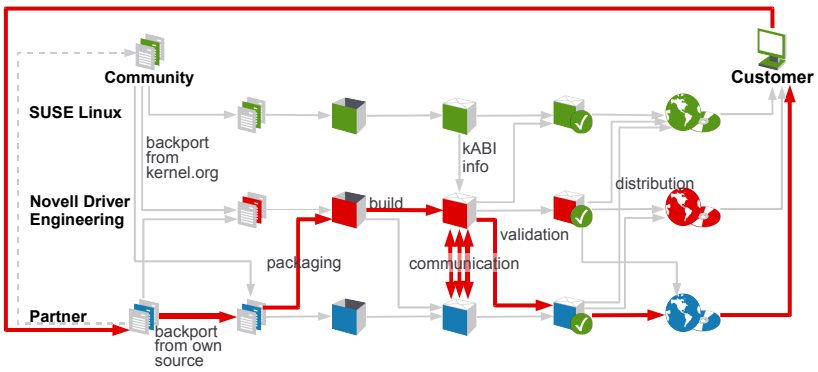
Initial build: Partner ports the driver to SUSE Linux kernel, packages, builds, validates and distributes. Partner provides Novell with kABI metadata.

Kernel updates: Novell sends notifications. Partner checks, eventually rebuilds and/or revalidates and distributes.

Driver updates: Partner updates its KMP package (for whichever reason, it is **its** driver!) partner builds, validates and distributes

(Eventually add **certification testing** for initial SUSE Linux release and service packs.)

Default for Security Sensitive Drivers



- Partner: backport and packaging
- Novell: build
- Partner: validation and distribution
- Partner and Novell: support
- Novell: kernel update notifications and rebuilds

For security sensitive drivers, we recommend the build service.

Initial build: Partner feeds backported KMP source package into build service. Novell builds the driver and provides it for validation and redistribution.

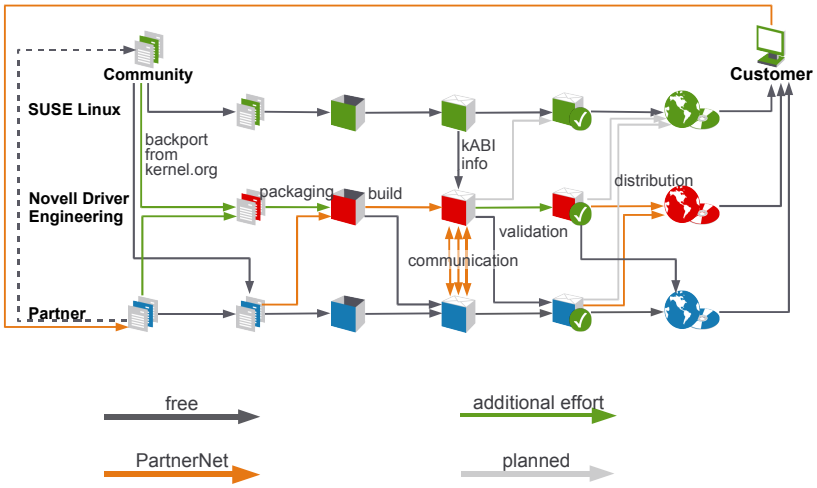
Kernel updates: Novell sends notifications, builds the driver, and provides it for validation and redistribution.

Driver updates: Partner provides updated driver KMP package (for whichever reason, it is **its** driver!) Novell builds the driver and provides it for validation and redistribution.

(Eventually add **certification testing** for initial SUSE Linux release and service packs.)

Partnership Participation Requirements

Driver Process: Participation

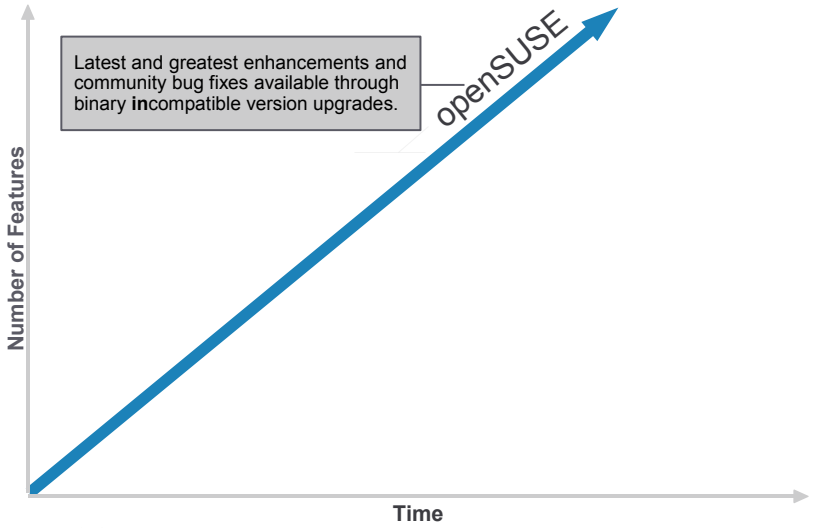


This diagram is color-coded to show what is covered by framework partnership contracts (black), what needs additional agreements (orange), and what needs dedicated case-by-case negotiations (green).

Maintenance: Backports of Fixes

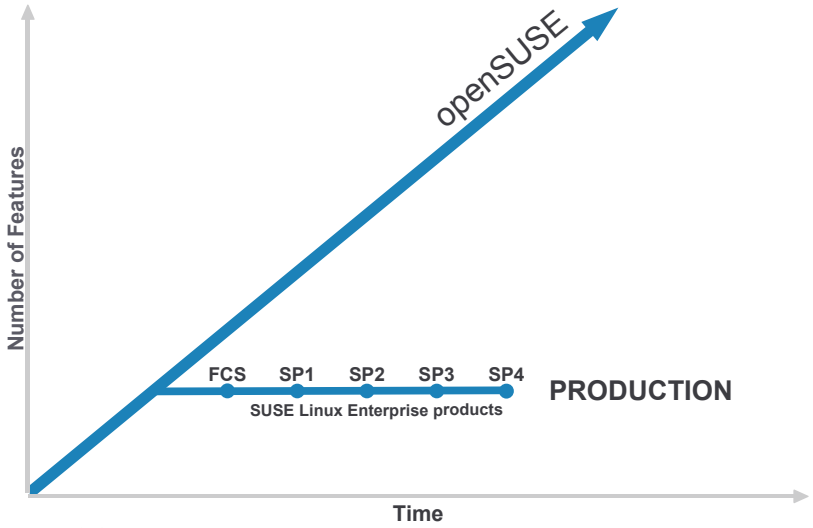
Why Kernel.org Integration Is Important in but Not Sufficient for Maintenance

Open Source Community:
Continuous Incompatible Rebuilds



The open source community is continuously updating. New features as well as bug fixes are done in the latest and greatest version.

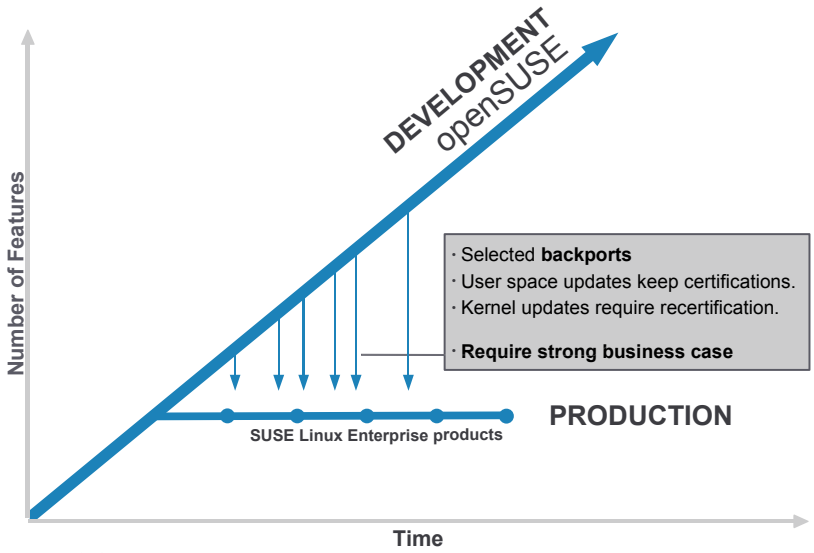
SUSE Linux Enterprise Products:
Stable User Space ABI



37 © Novell Inc. Confidential and Proprietary

Enterprise Linux is providing a stable platform based on one specific kernel snapshot.

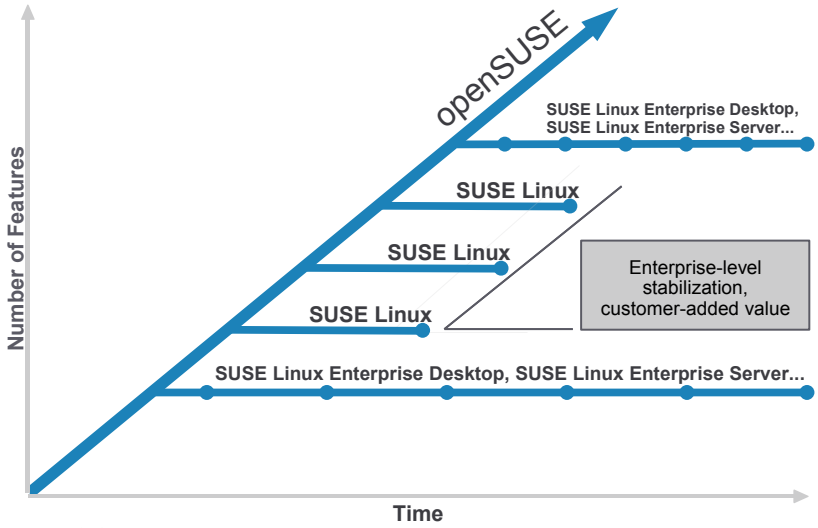
SUSE Linux Enterprise Products:
Only Selected Backports



Kernel engineering backports bug fixes to the specific enterprise kernel snapshots if the bug was already present before we forked off.

We backport only select feature enhancements and only to service packs (not between service packs) because such changes need to be validated. They may affect user space; feature enhancements trigger a full QA and recertification of applications.

SUSE Linux Enterprise Products:
Several Kernel Versions



The kernel changes significantly between the miscellaneous enterprise Linux versions.

A module backported to one version often doesn't compile on the next.

Building, Certifying and Distributing the Drivers

Build

- General SUSE build tool:
  - build.rpm
- Driver rpm specifics:
  - http://www.suse.de/~agruen/KMPM/
- Build service
  - Suggest drivers in bugzilla.novell.com
    - Category: Partner Linux Drivers
    - Product: Partner Linux Driver Package – Code 9 or Code 10
    - Partner prepares compliant packages
    - Novell prioritizes by business relationship
  - Partner controls what is released, when and where so (NDA)

build.rpm: autobuild tool, sandboxed builds on a developer workstation. Build target independent of the host-OS. e.g. On SUSE 10.1 developer workstation cross build for SLES9 or NLD9 or SLE10.

Driver packages: special package conventions to fit kernel and update mechanism.

Build service: autobuild system automatically rebuilding drivers for every kernel update.

Get Certified

- Component Certification
  - In place for LAN and storage adapters
  - In the works for graphics

Certification helps end customers understand that the driver and the operating system fit.

Distribute

- Distribution via Novell
  - Pure GPL only
  - Comply with export regulations
  - L3 relationship
  - Use build service
- Self-distribution
  - Create add-on product

There are two distribution options: Distribution via Novell and self distribution.

Novell will redistribute only pure GPL kernel modules. If Novell distribution is not an option, self distribution is technically enabled in our products.

Questions and Answers


Novell